BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problems Mailbox.

(51) Int. Cl.: G 06 f, 9/18

BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENTAMT

(52) Deutsche Kl.: 42 m3, 9/18

(10) (11) Offenlegungsschrift 2150506

(21) Aktenzeichen: P 21 50 506.6

(22) Anmeldetag: 9. Oktober 1971

(43) Offenlegungstag: 17. Mai 1973

Ausstellungspriorität: —

(30) Unionspriorität

(32) Datum: —

(33) Land: —

(31) Aktenzeichen: —

---

(54) Bezeichnung: Verfahren zur Herstellung von Querverbindungen zwischen Programmen von Datenverarbeitungsanlagen während der Programmausführung

(61) Zusatz zu: —

(62) Ausscheidung aus: —

(71) Anmelder: IBM Deutschland GmbH, 7000 Stuttgart

Vertreter gem. § 16 PatG: —

(72) Als Erfinder benannt: Ploechl, Wilhelm, 7031 Rohrau

DT 2150506

O 5.73 309 920/325 22/70

Böblingen, den 17. August 1971
km-sz

Amtliches Aktenzeichen: Neuanmeldung

Aktenzeichen der Anmelderin: Docket GE 971 501

Verfahren zur Herstellung von Querverbindungen zwischen Programmen von Datenverarbeitungsanlagen während der Programmausführung

Die Erfindung betrifft ein Verfahren zur Herstellung von Querverbindungen zwischen mehreren gleichzeitig im internen Speicher einer Datenverarbeitungsanlage gespeicherten Programmen während der Programmausführung, die unter der Regie eines Überwachungsprogrammes erfolgt.

Ein Programm zur Ausführung einer Datenverarbeitungsaufgabe mit Hilfe einer elektronischen Rechenanlage (Problemprogramm) besteht zumeist aus mehreren selbständigen Teilprogrammen, die an verschiedenen Stellen und/oder zu verschiedenen Zeiten geschrieben wurden. Jedes Teilprogramm, das auch Programmodul genannt wird, weist Befehle auf, durch die auf Teile anderer Programmodule Bezug genommen wird. Bei diesen Teilen kann es sich um Unterprogramme, Konstantwerte, Zwischenresultate usw. handeln, die in betreffenden anderen Programmoduln definiert sind und von diesen benutzt bzw. erzeugt werden. Die Bezugnahme kann über die echten Speicheradressen erfolgen, auf denen die entsprechenden Teile der anderen Programmodule gespeichert sind. Hierzu ist es aber erforderlich, daß die echten Speich radressen bereits be-

kannt sind, wenn die Teilprogramme geschrieben werden. Die Querverbindungen zwischen den einzelnen Teilprogrammen werden somit vom Programmierer endgültig festgelegt. Dies ist nur möglich, wenn die Speicherbelegung mit einer vorbestimmten Anzahl von Teilprogrammen fest vorausgeplant wird. Eine solche Arbeitsweise schränkt die Flexibilität in der Anwendung der Problemprogramme stark ein.

Die Bezugnahme von Befehlen eines Programmoduls auf Teile eines anderen Programmoduls kann ferner über Symbole erfolgen, die Namen der Programmteile darstellen, zu denen eine Querverbindung hergestellt werden soll. Dies erleichtert die Programmierung insofern, als der Programmierer sich nicht mit den echten Speicheradressen der Datenverarbeitungsanlage beschäftigen muß, auf denen die Programmteile später stehen werden. Durch ein Verbindungsprogramm werden die Symbole echten Speicheradressen zugeordnet und in die einzelnen Teilprogramme eingesetzt. Das Verbindungsprogramm erzeugt ein ladefähiges Problemprogramm, das in einem externen Speicher der Datenverarbeitungsanlage aufbewahrt wird, bis es für die eigentliche Programmausführung in den internen Speicher der Anlage geladen werden kann (IBM Systems Reference Library GC 2865342 "IBM System /360 Operating System - Introduction", Oktober 1969, Seiten 77 bis 79). Auf diese Weise wird die Programmierung erheblich erleichert und eine hohe Flexibilität bei der Anwendung erzielt. Es ist hierbei jedoch notwendig, daß vor der Verfügbarkeit des ladefähigen Problemprogramms ein separater Ablauf des Verbindungsprogrammes erfolgt, durch den die benötigten Querverbindungen zwischen den zur Zeit der Programmausführung gleichzeitig im internen Speicher der Datenverarbeitungsanlage befindlichen Teilprogrammen hergestellt werden. Ebenso ist jeweils ein neuer Lauf des Verbindungsprogrammes notwendig, wenn die Teilprogrammstruktur innerhalb des Problemprogrammes geändert werden soll. Dies erschwert insbesondere den Aufbau von solchen Problemprogrammen, innerhalb derer sich die Teilprogrammstruktur nicht nach einem vorgegebenen Plan, sondern in Abhängigkeit vom jeweiligen Bedarf

ändert (dynamische Programmstruktur). Während des Laufes des Verbindungsprogramms ist die Datenverarbeitungsanlage für die Ausführung von Problemprogrammen blockiert, weshalb dieser Lauf für den Benutzer eine Verlustzeit darstellt.

Aufgabe der Erfindung ist es, ein Verfahren anzugeben, das unter Vermeidung der genannten Nachteile eine Herstellung von Querverbindungen zwischen verschiedenen, zur gleichen Zeit im internen Speicher der Datenverarbeitungsanlage gespeicherten Programmen während der Programmausführung, also ohne vorausgehenden separaten Lauf eines Verbindungsprogrammes, gestattet. Erfindungsgemäß wird dies dadurch erreicht, daß in jedem Programm eine erste Tabelle von im Programm vorhandenen, programmextern definierten Symbolen mit einem zugehörigen, zunächst leeren Feld für die Adresse dieses Symbols gebildet wird, daß in jedem Programm eine zweite Tabelle von im Programm definierten, programmextern ansprechbaren Eingangssymbolen und den Adressen dieser Symbole gebildet wird, daß im Überwachungsprogramm eine Adressentabelle für die ersten und zweiten Tabellen der Programme erzeugt wird, daß beim Aufruf eines externen Symbols während des Ablaufes eines Programms ein Versuch gemacht wird, die Adresse dieses Symbols aus der ersten Tabelle des gleichen Programms in den betreffenden Befehl zu laden, daß eine Programmunterbrechung erfolgt, wenn die erste Tabelle für das betreffende Symbol noch keine Adresse enthält, daß aufgrund der Programmunterbrechung das Überwachungsprogramm wirksam wird und mit dem betreffenden externen Symbol alle im Speicher vorhandenen zweiten Tabellen nach einem gleichnamigen Eingangssymbol durchsucht, und daß die Adresse des ermittelten Eingangssymbols in das freie Feld des gleichnamigen externen Symbols der ersten Tabelle übertragen wird, wonach der Lauf des Programms an der Unterbrechungsstelle mit einem erneuten Versuch, die Adresse des externen Symbols aus der ersten Tabelle zu laden, fortgesetzt wird.

Weitere vorteilhafte Ausgestaltungen und Weiterbildungen der Er-

findung sind aus den Ansprüchen ersichtlich. Nachfolgend ist ein Ausführungsbeispiel der Erfindung an Hand von Zeichnungen erläutert. Es zeigen:

Fig. 1 — ein Blockdiagramm zur Veranschaulichung des bekannten Verfahrensablaufes, bei dem Querverbindungen zwischen verschiedenen Programmen mit Hilfe eines separaten Verbindungsprogrammes hergestellt werden,

Fig. 2 — eine Abwandlung des Blockdiagrammes von Fig. 1, wie sie sich durch Anwendung des erfindungsgemäßen Verfahrens ergibt,

Fig. 3 — ein vereinfachtes Ablaufdiagramm zur Darstellung des erfindungsgemäßen Verfahrens,

Fign. 4A, B — Strukturen der im Ablaufdiagramm von Fig. 3 verwendeten Verbindungs- und Adressentabellen,

Fign. 5A, B, C — den Stand der Verbindungs- und Adressentabellen von Fign. 4A, B bei der beispielsweisen Herstellung einer Querverbindung von einem Programm P1 zu einem Programm P2,

Fig. 6 — ein Ablaufdiagramm zur Bildung der Adressentabelle ADTAB im Überwachungsprogramm während des Ladens der Programme in den internen Speicher der Datenverarbeitungsanlage,

Fign. 7A, B — detailliertere Ablaufdiagramme für die Herstellung einer Verbindung zwischen zwei Programmen P1, P2,

Fign. 8A, B — ein Ablaufdiagramm für das Löschen der hergestellten Verbindungen, wenn eines der Programme

aus dem internen Speicher der Datenverarbeitungsanlage entfernt wird,

Fig. 9 ein Ablaufdiagramm für die zum Befehl LEAC gehörenden Mikroprogrammschritte und

Fig. 10 eine Darstellung des Formats des LEAC-Befehls.

Das Ablaufdiagramm von Fig. 1 zeigt den herkömmlichen Weg der Verknüpfung von drei Teilprogrammen zu einem ladefähigen Arbeitsprogramm. Jedes der Teilprogramme wird in einer geeigneten Programmiersprache geschrieben und danach in einen Lochkartenstapel 12 übertragen. Das im Lochkartenstapel enthaltene Programm wird als Quellenmodul QM bezeichnet. Jeder Quellenmodul QM1 bis QM3 wird in einem separaten Übersetzungsschritt 13 in einen Objektmodul OM übersetzt. Der Objektmodul ist ein arbeitsfähiges Teilprogramm, das die Operationsschlüssel und Adressen in der Sprache der Rechenanlage enthält. Der Übersetzungsschritt 13 erfolgt unter Anwendung geeigneter Übersetzungsprogramme TR-PRGR 1 bis 3. Die Objektmodule OM1 bis OM3 werden auf einem externen Speicher 14 der Rechenanlage zwischengespeichert. In einem weiteren Arbeitsschritt 15 werden die Objektmodule OM1 bis OM3 gemeinsam unter der Wirkung eines Verbindungsprogramms zu einem ladefähigen Objektprogramm LOP vereinigt, ~~der~~ das auf einem weiteren externen Speicher 16 zwischengespeichert wird. Das Verbindungsprogramm setzt in die Befehle der einzelnen Objektmodule, die auf Teile anderer Objektmodule Bezug nehmen, die relativen Adressen ein, die diesen Teilen in den anderen Objektmodulen zugewiesen sind. Vom Zwischenspeicher 16, der als Bibliotheksspeicher der Rechenanlage für ladefähige Objektprogramme dient, wird durch einen Ladeschritt 17 unter der Wirkung eines Ladeprogrammes das betreffende Objektprogramm LOP in den Hauptspeicher 18 der Rechenanlage geladen. Das im Hauptspeicher 18 stehende Programm LOP, dem durch das Ladeprogramm die echten Hauptspeicheradressen zugewiesen wurden, kann nun durch die Rechenanlage ausgeführt werden.

Das Ablaufdiagramm von Fig. 2 weicht gegenüber dem von Fig. 1 insofern ab, als der Schritt 15 und der externe Speicher 16 fehlen. Die als Kartenstapel 12 vorliegenden Quellenmodule QM1 bis QM3 werden im Schritt 13 durch Übersetzungsprogramme TR-PRGR 1 bis 3 zu Objektmodulen OM1 bis 3 übersetzt, welche als Lademodule LM1 bis 3 im externen Speicher 14 der Rechenanlage gespeichert werden. Der Speicher 14 dient als Bibliothek für ladefähige und arbeitsbereite Teilprogramme. Im Ladeschritt 17 werden durch das Ladeprogramm die Lademodule LM1 bis 3 nacheinander in den Hauptspeicher 18 der Rechenanlage geladen. Jeder Lademodul LM1 bis 3 erhält dabei seine eigenen Hauptspeicherplätze zugewiesen. Die in den Hauptspeicher geladenen Teilprogramme LM1 bis 3 weisen untereinander noch keine Querverbindungen auf. Die herzustellenden Querverbindungen sind lediglich symbolisch markiert. In jedem Teilprogramm sind die Symbole, die auf ein anderes der Teilprogramme Bezug nehmen, mit "EXTRN" bezeichnet. Ebenso sind in jedem Teilprogramm diejenigen Symbole, auf die von anderen der Teilprogramme Bezug genommen wird, mit "ENTRY" bezeichnet. Die im Schritt 13 zur Wirkung kommenden Übersetzerprogramme legen nach den in den Quellenmoduln QM enthaltenen Angaben am Anfang eines jeden Lademoduls LM1 bis 3 eine erste Tabelle aller EXTRN-Symbole und eine zweite Tabelle aller ENTRY-Symbole an. Außerdem wird beim Laden der Lademodule LM1 bis 3 in den Hauptspeicher eine Adressentabelle im Überwachungsprogramm gebildet für die Adressen aller ENTRY- und EXTRN-Tabellen. Die Rechenanlage kann nun mit der Programmausführung beginnen, wobei die im Hauptspeicher 18 stehenden Teilprogramme LM1 bis 3 gemeinsam das Objektprogramm bzw. Arbeitsprogramm bilden.

In Fig. 3 sind die wesentlichsten Schritte des erfindungsgemäßen Verfahrens zur Herstellung der Querverbindungen zwischen den im Hauptspeicher der Rechenanlage stehenden Programmen dargestellt. Während der Programmübersetzung werden in einem Schritt 21 am Beginn eines jeden Lademoduls eine erste Tabelle EXTAB und eine zweite Tabelle ENTAB gebildet. Die Struktur dieser Tabellen ist in Fig. 4A angegeben. Die Tabelle EXTAB weist eine Kopfzeile auf

mit den Eintragungen A und GEX. Die Eintragung A umfaßt ein Byte und stellt einen Indikator für das Vorhandensein einer Tabelle EXTAB dar. Wenn A = 1, schließen sich an die Kopfzeile Eintragungen der Tabelle EXTAB an. Wenn A den Wert O hat, enthält die Tabelle keine weiteren Eintragungen. Die Eintragung GEX umfaßt drei Bytes und gibt die Zahl der in der Tabelle EXTAB enthaltenen Elemente (Zeilen) an. Die folgenden Zeilen dieser Tabelle bestehen aus je einem Feld NAMEX und ADREX. Das erstgenannte Feld umfaßt acht Bytes und enthält den Namen eines im betreffenden Teilprogramm enthaltenen Symbols, das in einem anderen Teilprogramm definiert ist. Das Feld ADREX besteht aus drei Bytes, die sich anfangs im Null-Zustand befinden und später die Adresse des extern definierten Symbols aufnehmen. Die Tabelle EXTAB enthält so viele Zeilen, wie der Wert im Feld GEX angibt. Die Tabelle ENTAB enthält in einer Kopfzeile das Feld GEN, das aus drei Bytes besteht und die Anzahl der in dieser Tabelle enthaltenen Elemente bzw. Zeilen bezeichnet. Die folgenden Zeilen der Tabelle ENTAB enthalten je drei Felder: NAMEN, B, ADREN. Das Feld NAMEN besteht aus acht Bytes und enthält den Namen eines im betreffenden Programm definierten und von einem anderen Programm benützten Symbols. Derartige Symbole werden nachfolgend im allgemeinen mit Eingangssymbol bezeichnet. Das Feld B besteht aus einem Byte, das sich anfangs im Null-Zustand befindet und auf 1 gesetzt wird, wenn das zugehörige Feld ADREN benutzt wird. Das Feld ADREN besteht aus drei Bytes und enthält die Adresse des Symbols, das im zugehörigen Feld NAMEN eingetragen ist. Die schraffierten Felder beider Tabellen enthalten keine Information.

Im Schritt 21 von Fig. 3 werden somit für jedes Programm die Tabellen EXTAB und ENTAB angelegt. Die Eintragungen für die Felder NAMEX der Tabelle EXTAB und für die Felder NAMEN und ADREN für die Tabelle ENTAB werden aus den im betreffenden Programm enthaltenen Angaben gewonnen. In diesem Programm sind alle extern definierten Symbole mit der Bezeichnung EXTRN versehen, und für jedes der so gekennzeichneten Symbole erfolgt eine Eintragung in der Tabelle EXTAB. Hierbei bleiben die Felder ADREX dieser

Tabelle frei. Desgleichen sind im Programm die von anderen Programmen aufrufbaren Symbole mit ENTRY gekennzeichnet. Für jedes der so gekennzeichneten Symbole erfolgt bei der Übersetzung des Programms eine Eintragung in eines der Felder NAMEN der Tabelle ENTAB. Außerdem wird in das zu diesem Feld gehörende Feld ADREN die im Programm angegebene Adresse des betreffenden ENTRY-Symbols eingespeichert. Die Felder B bleiben hierbei jedoch im Null-Zustand.

Im folgenden Schritt 22, der während der Ladeoperation 17 von Fig. 2 zur Ausführung kommt, wird von dem Überwachungsprogramm, unter dessen Regie das Arbeitsprogramm in der Rechenanlage abläuft, eine Adressentabelle ADTAB für die Tabellen EXTAB und ENTAB hergestellt, die in den Teilprogrammen enthalten sind, welche zum Arbeitsprogramm gehören. Die Struktur der Tabelle ADTAB ist aus Fig. 4B ersichtlich. Jede Zeile dieser Tabelle weist folgende vier Felder auf: IDENT, E, AEX, F, AEN. Das Feld IDENT besteht aus acht Bytes und enthält den Namen eines in den Hauptspeicher 18 der Rechenanlage geladenen Programmes, das Tabellen EXTAB und/oder ENTAB besitzt. Das Feld E besteht aus einem Byte, das anfangs den Wert 0 enthält. Dieses Feld wird auf 1 gesetzt, wenn die Tabelle EXTAB in dem im Feld IDENT angegebenen Programm wenigstens ein EXTRN-Symbol enthält. Das Feld AEX umfaßt vier Bytes und enthält die Adresse der Tabelle EXTAB, die zum Programm gehört, dessen Name im zugehörigen Feld IDENT steht. Das Feld AEX enthält 0, wenn für das Programm keine Tabelle EXTAB existiert. Das Feld F besteht aus einem Byte, das anfangs den Wert 0 enthält. In diesem Feld wird der Wert 1 eingespeichert, wenn die Tabelle ENTAB, die zu dem im Feld IDENT angegebenen Programm gehört, während des Laufes des Arbeitsprogrammes wenigstens einmal benutzt wurde. Das Feld AEN besteht aus vier Bytes und enthält die Adresse der Tabelle ENTAB, die zum Programm gehört, dessen Name im Feld IDENT steht. Das Feld AEN befindet sich im Null-Zustand, wenn das betreffende Programm keine Tabelle ENTAB aufweist.

Die Tabelle ADTAB wird vom Überwachungsprogramm angelegt, das den Ablauf des Problemprogrammes in der Rechenanlage steuert. Das Überwachungsprogramm wird auch wirksam, um innerhalb des Problemprogrammes die erforderlichen Querverbindungen zwischen den im internen Speicher 18 stehenden, das Problemprogramm bildenden Teilprogrammen LM1 bis 3 herzustellen. Wenn während des Ablaufes des Problemprogrammes (Schritt 23 in Fig. 3) ein Befehl auftritt, der ein EXTRN-Symbol aufruft, wird versucht, die Adresse dieses Symbols aus der Tabelle EXTAB des Programmes in den betreffenden Befehl zu laden. Hierzu wird im Schritt 24 geprüft, ob sich in der Tabelle EXTAB für das betreffende EXTRN-Symbol bereits ein Adresseneintrag befindet. Dies geschieht unter Benutzung eines Befehls

$$\text{LEAC} \qquad R_1, D_2(X_2, B_2)$$

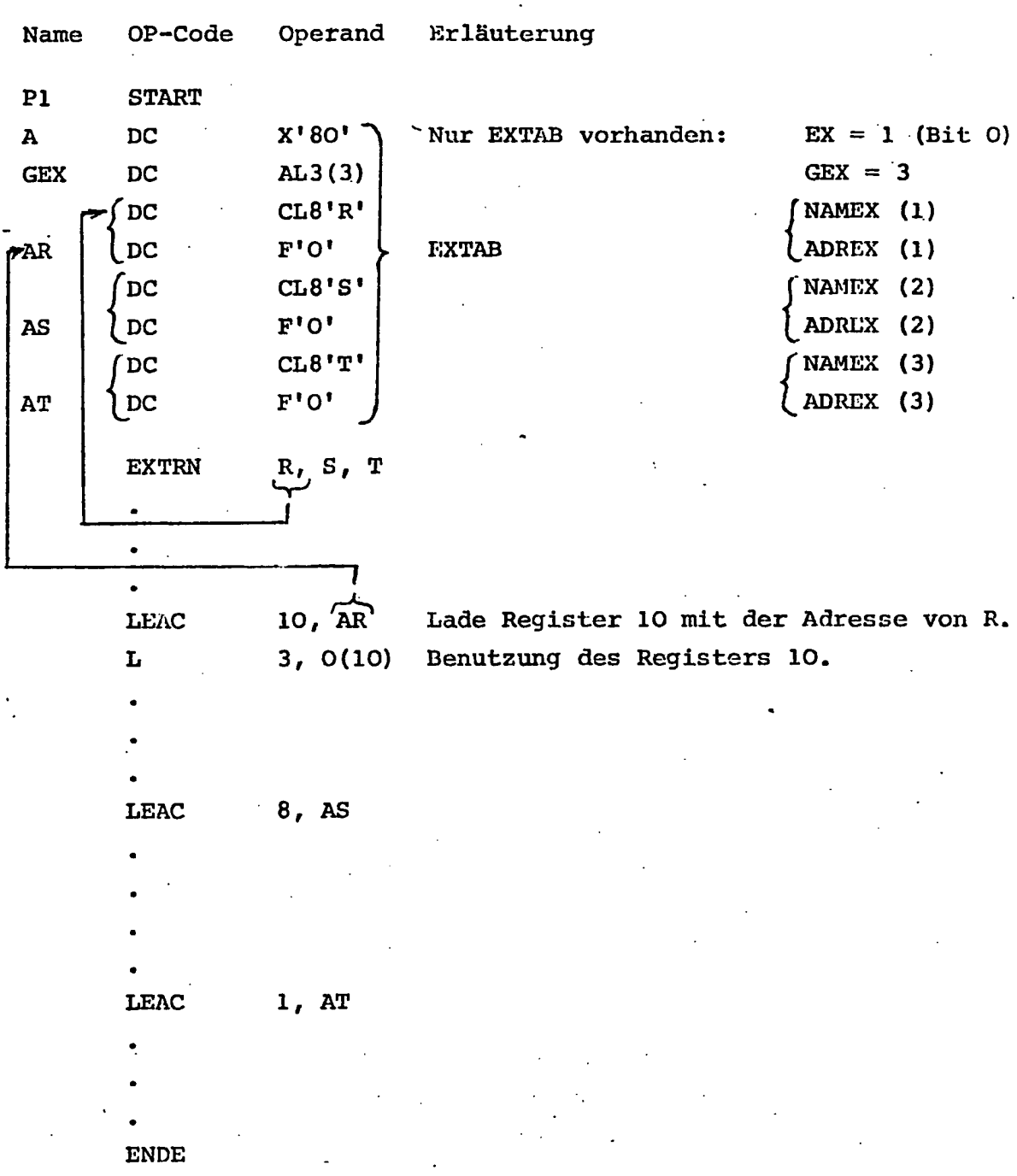
mit der Bezeichnung 'Adresse laden'. Hierin bedeutet LEAC den symbolischen Operationscode, R_1 eine Registeradresse als ersten Operanden des Befehls und $D_2(X_2, B_2)$ den zweiten Operanden, der eine Adresse darstellt, die sich aus dem Inhalt eines Indexregisters X_2, dem Inhalt eines Basisregisters B_2 und einer Verschiebeadresse D_2 zusammensetzt. Die vom Befehl ausgeführte Operation besteht darin, daß der zweite Operand in das vom ersten Operanden bezeichnete Register geladen wird. Mit einem derartigen Befehl können z. B. bekannte Datenverarbeitungsanlagen des IBM Systems /360, wie Modell 25, versehen werden durch Einspeicherung eines Mikroprogrammes, dessen Schritte im einzelnen in einem späteren Abschnitt in Verbindung mit Fig. 9 beschrieben werden.

Bei der Ausführung des Befehls LEAC wird geprüft, ob der im Programm als Adresse des betreffenden EXTRN-Symbols bezeichnete zweite Operand den Wert O besitzt. Wenn sich im Schritt 24 von Fig. 3 ergibt, daß die gesuchte Adresse einen von O abweichenden Wert hat, ist dies eine Aussage dafür, daß sich die Adresse in der Tabelle EXTAB befindet. Nach Übertragung der Adresse in das vom rsten Operanden bezeichnete Register wird deshalb der Programm-

lauf gemäß Schritt 23 fortgesetzt. Ergibt der Schritt 24 dagegen einen 0-Inhalt für die gesuchte Adresse, erfolgt gemäß Schritt 25 eine Unterbrechung des Programms und ein Aufruf des Überwachungsprogramms der Rechenanlage. Das Überwachungsprogramm durchsucht daraufhin alle verfügbaren Tabellen ENTAB mit dem betreffenden EXTRN-Symbol als Suchargument nach einem NAMEN-Feld gleichen Inhalts (Schritt 26). Hierzu benutzt das Überwachungsprogramm die Tabelle ADTAB, um für jedes Teilprogramm LM die zugehörige Tabelle ENTAB zu finden, deren Adresse in dem am weitesten rechts stehenden Feld AEN der Tabelle ADTAB gespeichert ist. Wenn beim Durchsuchen der ENTAB-Tabellen ein Feld NAMEN gefunden wird, dessen Inhalt gleich dem EXTRN-Symbol ist, wird der Inhalt des zugehörigen Feldes ADREN, das die Adresse des gleichnamigen ENTRY-Symbols enthält, in die Tabelle EXTAB des unterbrochenen Programms übertragen. Die Übertragung erfolgt im Schritt 27 (Fig. 3) in das Adressenfeld ADREX des EXTRN-Symbols, das die Unterbrechung verursacht hat. Nach Beendigung dieses Schrittes wird der Programmlauf gemäß Schritt 23 an der gleichen Stelle fortgesetzt, an der zuvor die Unterbrechung stattfand. Gemäß Schritt 24 wird wiederum mit einem LEAC-Befehl versucht, die Adresse des externen Symbols aus der Tabelle EXTAB zu erhalten. Da sich die gesuchte Adresse nun in dieser Tabelle befindet, hat der erneute Versuch Erfolg, und es erfolgt eine Fortsetzung des Programmlaufes unter Verwendung der Teile aus einem anderen Programmodul LM, die durch das EXTRN-Symbol bezeichnet wurden. Beim Auftreten des nächsten EXTRN-Symbols werden die Schritte 24 bis 27 in der erläuterten Weise wiederum durchlaufen.

An Hand der Fign. 5A bis C und der nachstehenden Codelisten wird nachfolgend zur Verdeutlichung der Schritte gemäß Fig. 3 ein Zahlenbeispiel erläutert. In diesem Beispiel wird davon ausgegangen, daß ein Programm P1 drei externe Symbole EXTRN R, S, T anspricht, die in einem Programm P2 definiert sind. Die angegebenen Codelisten basieren auf einer Rechenanlage des IBM Systems /360.

Code im Programm P1 (Codeliste 1)



```
Name    OP-Code    Operand     Erläuterung

P1      START
A       DC         X'80'       Nur EXTAB vorhanden:      EX = 1 (Bit 0)
GEX     DC         AL3(3)                                GEX = 3
        DC         CL8'R'                                NAMEX (1)
AR      DC         F'0'        EXTAB                     ADREX (1)
        DC         CL8'S'                                NAMEX (2)
AS      DC         F'0'                                  ADREX (2)
        DC         CL8'T'                                NAMEX (3)
AT      DC         F'0'                                  ADREX (3)

        EXTRN      R, S, T
        .
        .
        .
        LEAC       10, AR      Lade Register 10 mit der Adresse von R.
        L          3, 0(10)    Benutzung des Registers 10.
        .
        .
        .
        LEAC       8, AS
        .
        .
        .
        .
        LEAC       1, AT
        .
        .
        .
        ENDE
```

Das Übersetzungsprogramm (Schritt 13 von Fig. 2) baut aus der obigen das Programm P1 als Quellenmodul darstellenden Codeliste für jedes externe Symbol R, S_0 T ein Element der Tabelle EXTAB im Programm P1 auf. Diese Tabelle ist in Fig. 5A dargestellt. Sie ist auf Adresse 10 000 im Hauptspeicher 18 der Rechenanlage gespeichert und besteht aus drei Elementen, die in den NAMEX-Feldern (Fig. 4A) die Symbole R, S und T enthalten. Die ADREX-Felder dieser Elemente weisen vor Beginn des Programmlaufes alle den Wert 0 auf. Die mit b gekennzeichneten Felder enthalten keine signifikante Information. In der Kopfzeile der Tabelle EXTAB wird in die Bitstelle 0 des Feldes A die Markierung EX = 1 als Anzeige für das Vorhandensein einer Tabelle EXTAB eingestellt. Dies geschieht durch die Zeile 2 der Codeliste 1, indem die Definition einer hexadezimalen Konstante 80 die höchste Bitstelle des betreffenden Feldes auf den Wert 1 setzt. Durch die folgende Zeile der Codeliste 1 wird das Feld GEX in der Kopfzeile der Tabelle EXTAB auf den Wert 3 gestellt, da die Tabelle drei Elemente enthält. Die nächsten sechs Zeilen der Codeliste betreffen die Definition der Elemente R, S, T für die Tabelle EXTAB. Zur Vereinfachung der Darstellung wurde angenommen, daß das Programm P1 keine Tabelle INTAB aufweist.

Code im Programm P2 (Codeliste 2)

Name	OP-Code	Operand	Erläuterungen	
P2	START			
A	DC	X'40'	Nur ENTAB vorhanden:	EN = 1 (Bit 1)
GEX	DC	AL3(0)	Keine EXTAB-Elemente:	GEX = 0
	DC	X'00'	Feld nicht benützt:	0
GEN	DC	AL3(3)		GEN = 3
	DC	CL8'R'		NAMEN (1)
	DC	X'00'		B (1)
	DC	AL3(R)		ADREN (1)
	DC	CL8'S'	ENTAB	NAMEN (2)
	DC	X'00'		B (2)
	DC	AL3(S)		ADREN (2)
	DC	CL8'T'		NAMEN (3)
	DC	X'00'		B (3)
	DC	AL3(T)		ADREN (3)
	ENTRY	R, S, T,		
	.			
	.			
	.			
	.			
	.			
	.			
	.			
	.			
R	DS	F		
S	DC	CL100		
T	DS	D		
	.			
	.			
	.			
	ENDE			

geändert gemäß Eingabe
eingegangen am 26.6.72

Der obere Teil der Codeliste 2 dient zur Definition der Tabelle ENTAB im Programm P2 (Fig. 5A). Die Kopfzeile dieser Tabelle steht auf der Hauptspeicheradresse 16 000. Durch die Zeile 2 der Codeliste 2 wird mit Einsetzen einer hexadezimalen Konstante 40 in das Feld A dessen Bitstelle 1 auf den Wert 1 gesetzt, womit die Markierung EN = 1 vorgenommen wird. Dies bedeutet, daß das Programm P2 nur eine Tabelle ENTAB aufweist und keine Tabelle EXTAB. Dementsprechend wird durch die dritte Zeile der Codeliste 2 das Feld GEX der Kopfzeile auf 0 gesetzt. ~~In der zweite Zeile~~ der Tabelle ENTAB (Kopfzeile dieser Tabelle) ~~wird in das Feld~~ A eine 0 eingesetzt, da dieses Feld im folgenden nicht benutzt wird. Durch die fünfte Zeile der Codeliste 2 wird der Wert 3 für das Feld GEN der Tabelle ENTAB definiert. In den folgenden neun Zeilen der Codeliste 2 werden die drei Elemente R, S, T definiert. Hieraus folgt während der Übersetzung gemäß Schritt 13 von Fig. 2 ein Tabellenaufbau, wie ihn die Fig. 5A angibt. Beim Laden des übersetzten Programmes P2 gemäß Schritt 17 von Fig. 2 werden in die Adreßfelder ADREN der Elemente von Tabelle ENTAB die Hauptspeicheradressen 17 000, 17 004, 17 104 eingesetzt. Während der gleichen Operation, also während des Ladens, wird vom Überwachungsprogramm die Tabelle ADTAB angelegt, wie aus Fig. 5A ersichtlich ist. Hierbei wird in das dem Programmnamen P1 zugeordnete Adressenfeld AEX (Adresse von Tabelle EXTAB) die Hauptspeicheradresse 10 000 eingesetzt. Ebenso wird in das dem Programmnamen P2 zugeordnete Adressenfeld AEN (Adresse der Tabelle ENTAB) die Hauptspeicheradresse 16 004 eingesetzt. In Zeile 10 der Codeliste 1 sind die Symbole R, S, T als externe Symbole gekennzeichnet. Die folgenden Zeilen 11 bis 13 betreffen nicht dargestellte Teile des Programms P1. Die Zeile 14 enthält die Instruktion LEAC, durch welche die Adresse des externen Symbols R aus der Tabelle EXTAB in das Register 10 geladen werden soll. Diese Instruktion bewirkt während des Programmlaufes eine Programmunterbrechung, da das Adreßfeld des Elementes R in der Tabelle EXTAB beim ersten Auftreten dieser LEAC-Instruktion noch leer ist. Die während dieser Programmunterbrechung sich in den Tabellen ergebenden Änderungen sind aus Fig. 5B ersichtlich. Über

das zweite Element der Tabelle ADTAB findet das Überwachungsprogramm die Adresse 16 004 der Kopfzeile von Tabelle ENTAB des Programms P2. Das Überwachungsprogramm beginnt daraufhin eine Untersuchung der Felder NAMEN dieser Tabelle nach dem Symbol R und findet dieses bereits im ersten Element der Tabelle. Daraufhin wird die im Feld ADREN stehende Hauptspeicheradresse 17 000 in das Feld ADREX des ersten Elementes von Tabelle EXTAB des Programms P1 übertragen. Dieses Adressenfeld ist dem in der LEAC-Instruktion angegebenen externen Symbol R zugeordnet. Außerdem wird in das Feld B des ersten Elementes in der Tabelle ENTAB eine 1 eingesetzt zur Anzeige dafür, daß dieses Element während des Programmlaufes bereits benutzt wurde. In das Feld E der ersten Zeile der ADTAB wird eine 1 gespeichert als Anzeige dafür, daß in EXTAB von P1 mindestens eine Adresse gespeichert wurde. In das Feld F der zweiten Zeile der ADTAB wird ebenfalls eine 1 gespeichert als Anzeige dafür, daß die ENTAB von P2 mindestens einmal benutzt wurde. Das Programm P1 wird durch einen erneuten und diesmal erfolgreichen Versuch fortgesetzt, die Adresse des Symbols R aus der Tabelle EXTAB in das Register 10 zu laden. In der folgenden Zeile 15 wird diese aus dem Programm P2 geholte Adresse innerhalb des Programms P1 benutzt. Die Querverbindung vom Programm P1 zum Programm P2 wird somit während des Ablaufes des Programms P1 unter Zwischenwirkung des Überwachungsprogramms hergestellt. Entsprechende Querverbindungen bewirken die LEAC-Instruktionen in den Zeilen 19 und 24 der Codeliste 1. Die hierbei auftretenden Veränderungen in den Tabellen EXTAB von Programm P1 und ENTAB von Programm P2 sind in der aus Fig. 5B ersichtlichen Weise herleitbar. Die restlichen Zeilen der Codeliste 1 betreffen Teile des Programms P1, die für das Wesen der Erfindung ohne Bedeutung sind. Das gleiche trifft auf die Zeilen 16 bis 22 und 27 bis 29 der Codeliste 2 zu. In den Zeilen 24 bis 26 der Codeliste 2 sind die im Programm P2 von außen ansprechbaren Symbole R, S, T definiert.

Im oben beschriebenen Beispiel wird lediglich ein Zugriff in einer Richtung vom Programm P1 zum Programm P2 erläutert. Nach

geändert gemäß Eingabe
eingegangen am 26.6.72

den gleichen Prinzipien kann jedoch eine Verbindung in umgekehrter Richtung hergestellt werden, indem vom Programm P2 ein Zugriff zu Teilen des Programms P1 erfolgt. In diesem Falle erhält das Programm P2 an seinem Anfang die entsprechenden Definitionen zum Aufbau einer Tabelle EXTAB und das Programm P1 die entsprechenden Definitionen zum Aufbau einer Tabelle ENTAB. Die während des Programmlaufes wechselweise herstellbaren Querverbindungen sind auch nicht auf zwei Programme beschränkt. Sie können vielmehr auf eine beliebige Anzahl von gleichzeitig im Hauptspeicher 18 der Rechenanlage befindlichen Programmen ausgedehnt werden. Die Adressentabelle ADTAB (Fig. 5A), die zur Adressierung der Tabellen EXTAB und ENTAB in den verschiedenen Programmen benutzt wird, läßt sich, wie dargestellt, beliebig erweitern.

Unter Bezugnahme auf die Fign. 6 bis 8 werden nachfolgend die Schritte erläutert, die das Überwachungsprogramm der Rechenanlage ausführt, wenn das Programm in den Hauptspeicher geladen wird, wenn eine Querverbindung zu einem anderen, zur gleichen Zeit im Hauptspeicher befindlichen Programm hergestellt werden soll und wenn eines dieser Programme wieder aus dem Hauptspeicher entfernt wird. Die beim Laden eines Programmes vom Überwachungsprogramm auszuführenden Operationen sind aus Fig. 6 ersichtlich. Der Block 31 betrifft allgemein das Laden des Programms auf die Adresse ADR, wobei Operationen ausgeführt werden, die mit dem Gegenstand der Erfindung nicht in Verbindung stehen, wie z. B. die Zuweisung einer Hauptspeicheradresse als Anfangsadresse für das Laden des Programms. Im Schritt 32 wird geprüft, ob die am Anfang des Programms stehenden Tabellen EXTAB und ENTAB im Feld A (Fig. 4A) eine Anzeige EX = 0 und EN = 0 enthalten. Sind beide Anzeigen 0, erfolgt über 30 eine Rückkehr zum Ladeprogramm. das daraufhin die einzelnen Befehle des betreffenden Problemprogramms auf die zugewiesenen Hauptspeicherplätze überträgt. Enthält dagegen eine der ~~vier~~ EX und EN reservierten Bitstellen einen von 0 abweichenden Wert, wird im Schritt 33 untersucht, ob in der Adressentabelle ADTAB (Fig. 4B) noch Platz frei ist. Die Zahl der

in dieser Tabelle besetzten Elemente ist im Überwachungsprogramm durch das Symbol N erfaßt, während die Angabe MAX die maximal zulässige Elementenzahl für die Tabelle ADTAB bezeichnet. Ist die Bedingung N < MAX nicht erfüllt, so erfolgt über 34 eine Rückkehr ins Ladeprogramm, wobei in dem Programm, welches das Laden veranlaßt hatte, eine Anzeige gespeichert wird, daß aus Platzmangel keine Verbindung zu programmextern definierten Symbolen hergestellt werden kann. Im anderen Falle geht die Steuerung des Überwachungsprogrammes zum Schritt 35 über, der einen Laufwert i = 1 setzt. Der Laufwert i bezeichnet dasjenige Element der Tabelle ADTAB, das sich gerade in Behandlung befindet. Im Schritt 36 wird geprüft, ob das Element i der Tabelle ADTAB, im vorliegenden Falle also das erste Element dieser Tabelle, ein leeres IDENT-Feld enthält. Wenn das erste Element bereits besetzt ist, wird eine Nein-Antwort erhalten, worauf durch Schritt 37 eine Inkrementierung des Laufwertes i um 1 erfolgt. Der Schritt 38 prüft, ob die Tabellengröße bereits überschritten ist. Liefert der Schritt 38 eine Ja-Anzeige, so liegt ein Symstemfehler vor, der über 39 zur Anzeige gelangt. Befindet sich dagegen der Wert i innerhalb der vorgeschriebenen Tabellengröße, erfolgt eine Verzweigung zurück zum Schritt 36. Wenn der Schritt 36 anzeigt, daß das untersuchte IDENT-Feld leer ist, erfolgt im Schritt 40 eine Inkrementierung des die besetzten Elemente der Tabelle ADTAB anzeigenden Wertes N um 1. Im Schritt 41 wird daraufhin der Programmname des zu ladenden Programms in das Feld IDENT übertragen. Das Überwachungsprogramm prüft im Schritt 42, ob in der Tabelle EXTAB des zu ladenden Programms der EX-Anzeiger auf 1 steht. Ist dies der Fall, erfolgt durch Schritt 43 eine Übertragung der Adresse der Kopfzeile dieser Tabelle EXTAB in das Adressenfeld AEX des jeweils in Behandlung befindlichen Elementes der Tabelle ADTAB. Da diese Tabelle normalerweise am Anfang des zu ladenden Programms steht, handelt es sich dabei um die Anfangsadresse ADR dieses Programms. Daraufhin geht die Steuerung zum Schritt 44 über, der die Anzeige EN der Tabelle ENTAB des zu ladenden Programms auf das Vorliegen einer 1 prüft. Der Schritt 44 kommt auch zur Wirkung, wenn der Schritt 42 eine Nein-Anzeige erzeugt. Wird

für EN ein von 1 abweichender Wert festgestellt, liegt ein Systemfehler vor, der über 39 zur Anzeige gelangt. Im anderen Falle wird durch Schritt 45 das Adreßfeld des in Behandlung befindlichen Elementes in der Tabelle ADTAB geladen. Die zu ladende Adresse ergibt sich durch ADR + 4 * 12 * GEX. Dieser Ausdruck bedeutet, daß zur Anfangsadresse ADR des zu ladenden Programms eine Inkrementierung um den Speicherraum erfolgt, den die Tabelle ENTAB einnimmt. Die Tabelle ENTAB schließt sich somit unmittelbar an die Tabelle EXTAB an. Nach dem Schritt 45 erfolgt über 46 eine Rückkehr in das Ladeprogramm, welches daraufhin mit der Ladeoperation fortfährt.

An Hand der Fign. 7A und 7B wird die Herstellung von Querverbindungen zwischen gleichzeitig im Hauptspeicher der Rechenanlage befindlichen Programmen erläutert. Der Schritt 51 von Fig. 7A stellt den Aufruf eines extern definierten Symbols in einem von der Rechenanlage ausgeführten Programm dar. Wie an Hand der Codeliste 1 erläutert wurde, wird dieser Aufruf durch die Instruktion LEAC ausgeführt, welche die Adresse des in ihr enthaltenen zweiten Operanden in ein vom ersten Operanden bezeichnetes Register lädt. Innerhalb dieser Operation finden verschiedene Prüfschritte statt, von denen in Fig. 7A nur die Schritte 52, 53 dargestellt sind. Im Schritt 52 wird geprüft, ob die zu ladende Adresse des Operanden 2 den Wert 0 aufweist. Handelt es sich um einen von 0 abweichenden Wert, führt der Prüfschritt 53 eine Prüfung durch, ob es sich um eine geschützte Adresse handelt. Wird dies ebenfalls verneint, erfolgt nach Vornahme weiterer Prüfschritte eine Fortsetzung des Programms. Im Falle von Ja-Anzeigen erfolgt dagegen eine Programmunterbrechung durch den Schritt 54. Eine Ja-Anzeige im Schritt 52 bedeutet, daß die Adresse des gesuchten, extern definierten Symbols in der EXTAB-Tabelle des Programms nicht enthalten ist. Die Programmunterbrechung gemäß Schritt 54 hat eine Verzweigung zum Überwachungsprogramm zur Folge, das über den Schritt 56 die in Fig. 7B dargestellte Verbindungsroutine beginnt. Durch den Schritt 57 wird die Adresse Q des Operanden OP2 der Instruktion LEAC geholt.

Diese Adresse, die auf ein ADREX-Feld der Tabelle EXTAB des unterbrochenen Programms zeigt, wird im Schritt 58 um 8 vermindert, wodurch die Adresse des zu diesem Feld gehörenden Feldes NAMEX gefunden wird. Der Inhalt dieses Feldes, der dem aufgerufenen Symbol entspricht, wird im folgenden als Suchwert benutzt. Durch einen Schritt 59 wird ein Laufwert i = 1 gesetzt. Dieser Laufwert bezeichnet die Nummer des gerade angesprochenen Elements in der Adressentabelle ADTAB. Im Schritt 60 wird geprüft, ob das Feld IDENT des Elementes i in Tabelle ADTAB leer ist. Handelt es sich um ein leeres Feld, wird durch Schritt 61 eine Inkrementierung des Laufwertes i um 1 herbeigeführt. Der Schritt 62 prüft, ob der Laufwert i noch innerhalb der Größe der Tabelle ADTAB liegt. Ist dies der Fall, erfolgt eine Verzweigung zurück zum Schritt 60. Stellt das Überwachungsprogramm nach mehrmaliger Wiederholung der Schritte 60 bis 62 fest, daß die Tabellengröße überschritten ist, erfolgt über 63 eine Rückkehr ins Problemprogramm. Diesem wird hierbei angezeigt, daß kein Symbol mit dem gewünschten Namen im Hauptspeicher der Rechenanlage enthalten ist. Wenn der Schritt 60 ein Nein-Resultat liefert als Anzeige, daß das Element i der Tabelle ADTAB besetzt ist, wird ein Schritt 64 wirksam, durch den das dem gleichen Tabellenelement angehörende Adressenfeld AEN geprüft wird, ob sein Inhalt von 0 abweicht. Hierdurch wird festgestellt, ob das Programm, dem dieses Element über das IDENT-Feld zugeordnet ist, eine Tabelle ENTAB enthält. Liefert der Schritt 64 eine Ja-Anzeige, wird durch den Schritt 65 ein Laufwert J = 1 gesetzt, der das jeweils in Behandlung befindliche Element der betreffenden Tabelle ENTAB bezeichnet. Andernfalls erfolgt vom Schritt 64 eine Verzweigung zum Schritt 61. Das Überwachungsprogramm prüft nun durch eine Suchschleife, der die Schritte 66, 67 und 68 angehören, ob der Name des gesuchten Symbols in der Tabelle ENTAB enthalten ist. Der Schritt 66 prüft den Namen des Elementes J auf Gleichheit mit dem im Schritt 58 definierten Suchwert. Ein Nein-Resultat dieses Vergleiches führt zu einer Inkrementierung des Laufwertes J um 1 durch den Schritt 67. Im Schritt 68 wird geprüft, ob der Laufwert noch innerhalb der Größe der Tabelle ENTAB liegt. Ist dies der Fall, wird eine

Wiederholung des Schrittes 66 eingeleitet, andernfalls erfolgt eine Verzweigung zum Schritt 61, wodurch zur Prüfung der Tabelle ENTAB eines anderen Programmes übergegangen wird.

Wenn der Schritt 66 eine Anzeige liefert, daß der Inhalt des NAMEN-Feldes mit dem Suchwert identisch ist, wird der Schritt 69 ausgeführt, der den Inhalt des Feldes ADREN im Element J der durchsuchten ENTAB-Tabelle in das Feld ADREX überträgt, das auf Adresse Q steht und dem Suchwert (EXTRN-Symbol) als Adressenfeld zugeordnet ist. Im Schritt 69 wird auch des Feld E des noch aufgerufenen Elementes i in der Tabelle ADTAB auf 1 gestellt, um dadurch zu markieren, daß in der zu diesem Element gehörenden Tabelle EXTAB wenigstens eine Adresse gespeichert wurde. Als nächstes wird der Schritt 70 ausgeführt, durch den das Element B in der durchsuchten Tabelle ENTAB auf 1 gestellt wird zur Anzeige dafür, daß das Adressenfeld ADREN dieses Elementes benützt wird. Außerdem wird durch den Schritt 70 das Feld F des Elementes i in der Adressentabelle ADTAB auf 1 gesetzt zur Anzeige dafür, daß die diesem Element zugeordnete Tabelle ENTAB mindestens einmal benutzt wurde. Über den Schritt 71 erfolgt nun eine Rückkehr in das Problemprogramm. Durch die oben erläuterten Schritte wurde die Adresse des externen Symbols in die Tabelle EXTAB gebracht. Das Problemprogramm wird durch eine Wiederholung der LEAC-Instruktion, die zuvor die Unterbrechung verursacht hatte, fortgesetzt.

Wenn eines der im Hauptspeicher 18 der Rechenanlage enthaltenen Programme, zwischen denen während der Programmausführung Querverbindungen hergestellt wurden, aus dem Hauptspeicher entfernt werden soll, müssen die in den im Speicher verbleibenden Programmen enthaltenen Querverbindungen aufgelöst werden, um Adressierungsfehler während der weiteren Ausführung der Programme zu vermeiden. Die hierzu notwendigen Arbeitsschritte sind in den Fign. 8A und 8B dargestellt. Mit einem Schritt 76 sind diejenigen Operationen des Überwachungsprogramms angegeben, die eine Entfernung des auf der Adresse ADR geladenen Programms aus dem Hauptspeicher 18 bewirken und die nicht Gegenstand der Erfindung

sind. Durch den Schritt 77 wird der Laufwert i für die Arbeit an der Adressentabelle ADTAB auf 1 gestellt. Im Schritt 78 wird das Feld IDENT im Element i dieser Tabelle aufgerufen und geprüft, ob sein Inhalt mit dem Namen des zu entfernenden Programms identisch ist. Wenn dies nicht der Fall ist, erfolgt eine Inkrementierung des Laufwertes i um 1. Hierzu dient der Schritt 79, der durch einen Prüfschritt 80 gefolgt wird, welcher feststellt, ob i noch innerhalb der Tabellengröße liegt. Ist dies der Fall, erfolgt eine Wiederholung des Schrittes 78 und gegebenenfalls auch der Schritte 79 und 80. Wird hierbei die Tabelle ADTAB aufgearbeitet, ohne daß sich ein Feld IDENT mit dem Namen des zu entfernenden Programms findet, wird die Verbindung-lösen-Routine über 114 beendet. Das Programm hat in diesem Fall weder eine Tabelle EXTAB noch eine Tabelle ENTAB. Durch eine Übereinstimmungsanzeige des Schrittes 78 wird der Schritt 82 wirksam, der eine Löschung des ermittelten IDENT-Feldes veranlaßt. Der Kennwert N für die Zahl der in der Tabelle ADTAB besetzten Elemente wird im Schritt 83 um 1 reduziert. Der Schritt 84 prüft, ob der Wert von N < O ist. Ein bejahendes Ergebnis dieser Prüfung führt zu einer Systemfehler-Anzeige über 85. Andernfalls erfolgt durch Schritt 86 eine Prüfung des Feldes AEN im Element i der Tabelle ADTAB, ob dieses Feld einen von O abweichenden Wert enthält. Ein Nein-Resultat dieses Prüfschrittes führt über 115 zum Ende der Verbindung-lösen-Routine, da in diesem Falle das Programm keine ENTAB-Tabelle hatte. Ein Ja-Resultat des Schrittes 86 führt zum Schritt 87, welcher prüft, ob das Feld F des Elementes i einen Inhalt hat, der von O abweicht. Auch hier bewirkt ein Nein-Resultat über 116 eine Beendigung der Verbindung-lösen-Routine, da in diesem Falle die dem Element i zugeordnete Tabelle ENTAB nicht benutzt worden ist. Ein Ja-Resultat der Prüfung von Schritt 87 führt zum Schritt 88, der den Laufwert J auf 1 setzt, wobei für die folgende Adressierung der Elemente der Tabelle ENTAB die Adresse AEN (i) als Basis genommen wird. Der Schritt 89 prüft, ob das Feld B des Elementes J in der angesprochenen Tabelle ENTAB einen von O abweichenden Wert enthält. Ein Nein-Resultat dieser Prüfung zeigt an, daß zu dem ENTRY-Symbol, das dem betreffenden

Tabellenelement zugeordnet ist, keine Verbindung hergestellt wurde. Durch den Schritt 90 erfolgt eine Inkrementierung des J-Wertes um 1 und der Schritt 91 prüft, ob der inkrementierte J-Wert noch innerhalb der Größe der behandelten Tabelle ENTAB liegt. Ist dies der Fall, wird der Schritt 89 und gegebenenfalls auch die Schritte 90 und 91 wiederholt. Andernfalls wird durch einen Schritt 92 das gesamte ADTAB-Element i gelöscht, wonach die Verbindung-lösen-Routine über 117 beendet wird.

Wenn der Schritt 89 ein Ja-Resultat als Anzeige dafür liefert, daß das Element J bereits Gegenstand einer Querverbindung war, wird der Schritt 94 (Fig. 8B) wirksam, der einen neuen Laufwert L = 1 einstellt. Dieser Laufwert wird zur Behandlung der IDENT-Felder der Tabelle ADTAB benutzt. Ein Schritt 95 prüft, ob das Feld IDENT des Elementes L in dieser Tabelle leer ist. Eine Ja-Anzeige hierfür führt zu einer Inkrementierung des Wertes L um 1 im Schritt 96. Der Schritt 97 prüft, ob der Wert L noch innerhalb der Größe der Tabelle ADTAB liegt. Wenn dies der Fall ist, erfolgt eine Verzweigung zurück zum Schritt 95. Andernfalls verzweigt das Überwachungsprogramm zum Schritt 90 von Fig. 8A, der den Laufwert J für die Behandlung der angesprochenen ENTAB-Tabelle inkrementiert. Ergibt sich bei der Durchführung des Schrittes 95, daß das Feld IDENT des Elementes L in der Tabelle ADTAB nicht leer ist, wird ein Prüfschritt 102 wirksam, der durch Prüfung des Feldes AEX im Element L auf einen von 0 abweichenden Wert feststellt, ob für dieses Element eine Tabelle EXTAB existiert. Ein Ja-Resultat dieses Schrittes führt zu einem weiteren Prüfschritt 103, der durch Untersuchung des Feldes E im gleichen Element feststellt, ob bereits eine Adresse in dieser EXTAB-Tabelle gespeichert worden ist. Nein-Resultate der Schritte 102 und 103 bewirken jeweils eine Verzweigung zum Schritt 96, der den Laufwert L inkrementiert. Als Folge eines Ja-Resultates des Schrittes 103 gelangt die Steuerung zum Schritt 104, der einen weiteren Laufwert K = 1 definiert und einen Anzeigeschalter AZSCH auf das Prüfkriterium 0 setzt. Der Laufwert K dient zur Behandlung der Elemente in der angesprochenen EXTAB-Tabelle, und der Schalter

geändert gemäß Eingabe
eingegangen am 26.6.72

wird beim Abfragen der Adressenfelder ADREX dieser Tabelle benutzt. In den folgenden Schritten wird mit dem Namen des ENTAB-Elementes, dessen B-Feld einen von 0 abweichenden Wert hat (Schritt 89 in Fig. 8A), eine Suche durch alle vorhandenen Tabellen EXTAB durchgeführt. Der Name des ENTAB-Elementes wird dabei als Suchargument benutzt. Wird ein gleichnamiger Inhalt in einem Feld NAMEX der Tabelle EXTAB gefunden, so wird das dazugehörige Adressenfeld gelöscht. Diese Operationen beginnen mit Schritt 105, der den Inhalt des Feldes NAMEN im Element J der Tabelle ENTAB des aus dem Speicher zu entfernenden Programms mit dem Inhalt des Feldes NAMEX im Element K der in Behandlung befindlichen Tabelle EXTAB vergleicht. Ergibt diese Vergleichsoperation eine Übereinstimmung, so wird durch Schritt 106 das zum Feld NAMEX gehörige Adressenfeld ADREX gelöscht. Der folgende Schritt 107 inkrementiert den Wert K um 1, wonach im Schritt 108 geprüft wird, ob der K-Wert noch innerhalb der Größe der in Behandlung befindlichen EXTAB-Tabelle liegt. Wenn dies der Fall ist, wird der Schritt 105 wiederholt. Wenn der Schritt 105 eine Nein-Aussage liefert, erfolgt durch Schritt 109 eine Prüfung, ob das Adressenfeld ADREX des Elementes K in der Tabelle EXTAB den Wert 0 enthält. Wenn dies der Fall ist, wird als nächstes der Schritt 107 ausgeführt, andernfalls wird durch einen Schritt 110 der Anzeigeschalter AZSCH in den Zustand 1 gebracht. Hierdurch wird eine Anzeige gespeichert, daß das letzte behandelte Element K der Tabelle EXTAB eine Eintragung enthält. Nach Beendigung des Schrittes 110 wird der Schritt 107 ausgeführt. Wenn die Operation gemäß Schritt 108 zu einem Ja-Resultat führt, wird durch einen Schritt 111 abgefragt, ob sich der Schalter AZSCH im 1-Zustand befindet. Ist dies der Fall, so befindet sich in der untersuchten Tabelle EXTAB wenigstens noch ein belegtes Adressenfeld ADREX, dessen zugeordnetes Feld NAMEX nicht mit dem als Argument benutzten Inhalt des Feldes NAMEN im Element J der Tabelle ENTAB in dem aus dem Speicher zu entfernenden Programm übereinstimmt. Liefert der Schritt 111 die Anzeige "nein", so wird durch einen Schritt 112 das Feld E des Elementes L in der Adressentabelle ADTAB auf 0 gesetzt. Auf die Schritte 111 und 112 folgt jeweils eine Verzweigung zum

Schritt 96, um eine Weiterschaltung auf das nächste Element der Adressentabelle vorzunehmen.

Durch die obigen Verfahrensschritte wurden alle Querverbindungen gelöst, die von anderen Programmen zu dem aus dem Hauptspeicher der Rechenanlage zu entfernenden Programm bestanden haben. Die EXTAB-Tabellen der anderen Programme wurden auf derartige Querverbindungen durchgesucht, und die gefundenen Querverbindungen wurden gelöscht (Schritt 106). Außerdem wurde das dem zu entfernenden Programm zugeordnete Element der Adressentabelle ADTAB ermittelt und ebenfalls gelöscht (Schritte 82 und 92). Für das oben erläuterte Zahlenbeispiel ist das Ergebnis dieser Operationen in Fig. 5C dargestellt. Hierbei wurde davon ausgegangen, daß das Programm P2 aus dem Hauptspeicher zu entfernen ist und daß die Querverbindungen, die von dem im Hauptspeicher bleibenden Programm P1 zum Programm P2 bestehen, gelöst werden sollen.

An Hand der Fig. 9 werden im einzelnen die Operationsschritte erläutert, die zur Ausführung des Befehls LEAC bei einer Datenverarbeitungsanlage des IBM Systems /360 durchgeführt werden. Es kann sich hierbei z. B. um eine mit einem Mikroprogramm arbeitende Anlage des Typs IBM System /360, Modell 25, handeln.

Der Ausführung des LEAC-Befehls voraus geht der Interpretationszyklus (I-Zyklus), der sich vom Interpretationszyklus anderer Befehle des gleichen Typs nicht unterscheidet. Durch einen Schritt 121 werden die Bytes 0 und 1 des zu interpretierenden Befehls aus dem Hauptspeicher 18 der Anlage in ein nicht dargestelltes Befehlsregister gelesen. Danach erfolgt durch Schritt 122 eine Inkrementierung der Befehlsadresse um 2. Im Byte 0 des Befehls ist dessen Operationscode enthalten. Der Schritt 123 prüft, ob der Befehl des gelesenen Operationscodes ein Befehl vom Typ RX ist. Da der LEAC-Befehl diesem Befehlstyp angehört, ist der Verneinungsfall, der gemäß Block 124 eine Abfrage nach dem Vorliegen anderer Befehlstypen vorsieht, für die vorliegende Betrachtung ohne Interesse. Wenn der Schritt 123 eine Ja-Anzeige

geändert gemäß Eingabe
eingegangen am 26.6.72

liefert, werden durch den Schritt 125 die Bytes 2 und 3 des Befehls gelesen, wonach wiederum eine Inkrementierung der Befehlsadresse um 2 durch den Schritt 126 erfolgt. Im Schritt 127 wird das vom zu interpretierenden Befehl bezeichnete Basisregister daraufhin untersucht, ob es den Wert 0 enthält. Es handelt sich dabei um das Basisregister, das durch die Komponente B_2 im LEAC-Befehl (Fig. 10) bezeichnet wird. Diese Komponente befindet sich auf den Bitstellen 16 bis 19 des Befehls, die von der ersten Hälfte des Bytes 2 eingenommen werden. Ist der Inhalt des Basisregisters 0, dann wird durch Schritt 128 festgestellt, daß die Adresse des zweiten Operanden gleich der im Adressenfeld D_2 enthaltenen Verschiebeadresse ist. Im anderen Falle setzt der Schritt 129 die Adresse des zweiten Operanden gleich dem Inhalt des Basisregisters B_2 plus der Verschiebeadresse D_2. In beiden Fällen wird als nächstes der Schritt 130 ausgeführt, durch den das Indexregister X_2 auf das Vorliegen eines Null-Inhaltes geprüft wird. Das Indexregister X_2 wird durch die zweite Hälfte des Bytes 1 in den Bitstellen 12 bis 15 des LEAC-Befehls bestimmt. Eine Ja-Anzeige hat zur Folge, daß der Schritt 131 die durch den Schritt 128 bzw. 129 ermittelte Adresse um den Inhalt des Indexregisters erhöht, während eine Nein-Anzeige des Schrittes 130 die Steuerung veranlaßt, den Schritt 131 zu umgehen. Damit liegt nun die Adresse des zweiten Operanden der LEAC-Instruktion fest, und es folgt nun in einer Anzahl von Prüfschritten eine Prüfung dieser Adresse in verschiedener Hinsicht.

Der Schritt 132 prüft, ob es sich um eine gültige Adresse handelt, d. h. ob die Adresse innerhalb des zulässigen Adressenbereiches der Datenverarbeitungsanlage liegt. Wenn dies nicht der Fall ist, erfolgt über 133 die Anzeige eines Adressenfehlers vom Typ I. Liegt eine gültige Adresse vor, wird durch einen Schritt 134 festgestellt, ob der zweite Operand auf einer Vollwortgrenze beginnt. Auch hier liegt ein Adressierfehler vor, wenn der Schritt 133 eine Nein-Anzeige zum Ausgang 135 liefert. Diese Art von Adressierfehler ist mit II klassifiziert. Eine Ja-Anzeige führt zu einem weiteren Prüfschritt 136, durch den festgestellt wird,

ob die ermittelte Adresse eine geschützte Adresse ist. Ist dies der Fall, erfolgt über 137 die Anzeige eines Adressierfehlers vom Typ III, der eine Speicherschutz-Unterbrechung des laufenden Programms veranlaßt. Im anderen Falle wird als nächstes der Schritt 138 ausgeführt, welcher den Operationscode des Befehls (Bitstellen 0 bis 7) prüft, ob er den Inhalt "LEAC" aufweist. Wenn dies nicht der Fall sein sollte, wird bei 139 durch im einzelnen nicht dargestellte Operationen eine Abfrage nach dem Vorliegen anderer OP-Codes durchgeführt. Dies ist jedoch hier ohne Interesse, da von der Voraussetzung ausgegangen wird, daß der zu interpretierende Befehl ein LEAC-Befehl ist. Mit der Ja-Anzeige aus Schritt 138 ist der Interpretationszyklus beendet, und die Steuerung tritt in den Ausführungszyklus (E-Zyklus) ein. Der erste Schritt dieses Zyklus besteht im Lesen des zweiten Operanden im Schritt 140. Durch Schritt 141 wird der zweite Operand daraufhin geprüft, ob er den Wert 0 aufweist. Liegt ein von 0 abweichender Wert vor, so wird mit dem Schritt 142 der zweite Operand in das Register R1 übertragen, das den ersten Operanden der LEAC-Operation darstellt. Damit ist die Operation "Adresse laden" ausgeführt, und die Maschine kann mit der Interpretation des folgenden Befehls beginnen. Liefert dagegen der Schritt 141 eine Ja-Anzeige, so wird durch einen Schritt 143 ein Unterbrechungscode 0020 in ein dafür vorgesehenes, nicht dargestelltes Register gebracht, was in der Folge bei 144 eine Unterbrechung des laufenden Programms zu Folge hat. Die Programmunterbrechung gemäß Schritt 144 ist mit der Programmunterbrechung von Schritt ~~155~~ 54 in Fig. 7A identisch. Sie bewirkt einen Aufruf des Überwachungsprogramms zur Ausführung der oben an Hand der Fig. 7B erläuterten Operationen.

Die vorausgehend an Hand der Fig. 9 beschriebenen Operationsschritte sind durch ein entsprechendes Mikroprogramm auf bekannten mikroprogrammierten Maschinen des IBM Systems /360 ausführbar. Eine Anlage dieses Typs ist z. B. in der deutschen Offenlegungsschrift 1 815 078 im Detail beschrieben.

Beim oben beschriebenen Ausführungsbeispiel wurde davon ausgegangen, daß die Tabellen EXTAB und ENTAB jeweils in dem Speicher gebildet werden, in welchem sich das betreffende Problemprogramm

befindet. Im Beispiel ist dies normalerweise der Hauptspeicher 18. Sofern es sich bei diesem Speicher nicht um einen sehr schnellen Speicher handelt, ist es im Interesse einer hohen Arbeitsgeschwindigkeit vorteilhaft, diese Tabellen in einen oder mehreren besonders schnellen Arbeitsspeichern unterzubringen, die durch die Befehle, welche auf ENTRY- oder EXTRN-Symbole Bezug nehmen, ansteuerbar sind. Das gleiche gilt für die Adressentabelle ADTAB, wobei die Ansteuerung des betreffenden Arbeitsspeichers durch das Überwachungsprogramm erfolgt. Datenverarbeitungsanlagen, die neben dem Hauptspeicher schnelle Arbeitsspeicher bzw. allgemein verwendbare Registersätze aufweisen, sind allgemein bekannt, wie beispielsweise die obengenannte Offenlegungsschrift zeigt.

Das erfindungsgemäße Verfahren ist auch bei Rechenanlagen ausführbar, welche über eine Instruktion zum Laden extern definierter Adressen nach Art der Instruktion LEAC nicht verfügen. In diesem Falle ist in die Codelisten (z. B. Codeliste 1) anstelle der Instruktion LEAC die folgende Befehlsequenz einzufügen:

```
L       10, = A(EX)
LTR     10, 10
BNZ     *+ 6
SVC     nnn
```

Diese Befehle, die z. B. zum Standardbefehlsatz von Datenverarbeitungsanlagen des IBM System /360 gehören, haben folgende Bedeutung. Durch den Befehl L wird eine im zweiten Operanden näher spezifizierte Adresse in ein durch den ersten Operanden angegebenes Register geladen. Die zweite Instruktion hat die Bedeutung "Laden und Testen". Hierdurch wird der Inhalt des Registers 10 wiederum in das Register 10 geladen, wobei verschiedene Testoperationen zur Ausführung gelangen, die sich auf den zu ladenden Zahlenwert beziehen. Eine dieser Testoperationen ist die Prüfung, ob der zu ladende Wert 0 ist. Der Befehl BNZ betrifft eine Verzweigung, wenn der geprüfte Wert von 0 abweicht. Hierdurch wird der nächste Befehl übersprungen, wenn der in der Instruktion LTR getestete Wert von 0 abweicht. Ist dies nicht der Fall, kommt die Instruktion SVC zur Ausführung, die eine Verzweigung zum Über-

wachungsprogramm ausführt. Die von diesen Befehlen ausgeführten Operationen entsprechen somit in ihrer Auswirkung auf das erfindungsgemäße Verfahren der erläuterten Instruktion LEAC.

PATENTANSPRÜCHE

1. Verfahren zur Herstellung von Querverbindungen zwischen mehreren gleichzeitig im internen Speicher einer Datenverarbeitungsanlage gespeicherten Programmen während der Programmausführung, die unter der Regie eines Überwachungsprogramms erfolgt, dadurch gekennzeichnet, daß zu jedem Programm eine erste Tabelle von im Programm vorhandenen, programmextern definierten Symbolen mit einem zugehörigen, zunächst leeren Feld für die Adresse dieses Symbols gebildet wird, daß zu jedem Programm eine zweite Tabelle von im Programm definierten, programmextern ansprechbaren Eingangssymbolen und den zugehörigen Adressen dieser Symbole gebildet wird, daß zum Überwachungsprogramm eine gemeinsame Adressentabelle für die ersten und zweiten Tabellen der Programme erzeugt wird, daß beim Aufruf eines externen Symbols während des Ablaufes eines Programms ein Versuch gemacht wird, die Adresse dieses Symbols aus der ersten Tabelle des gleichen Programms für den betreffenden Befehl zu laden, daß eine Programmunterbrechung erfolgt, wenn die erste Tabelle für das betreffende Symbol noch keine Adresse enthält, daß aufgrund der Programmunterbrechung das Überwachungsprogramm wirksam wird und mit dem betreffenden externen Symbol alle im Speicher vorhandenen zweiten Tabellen nach einem gleichnamigen Eingangssymbol durchsucht, und daß die Adresse des ermittelten Eingangssymbols in das freie Feld des gleichnamigen externen Symbols der ersten Tabelle übertragen wird, wonach der Lauf des Programms an der Unterbrechungsstelle mit einem erneuten Versuch, die Adresse des externen Symbols aus der ersten Tabelle zu laden, fortgesetzt wird.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß die zweite Tabelle für jedes aus Eingangssymbol und zugehöriger Adresse bestehendes Element ein Markierungsfeld

aufweist, in das ein Benutzungskennzeichen eingespeichert wird, wenn das betreffende Element zum erstenmal zum Aufbau einer Querverbindung benützt wird, und daß die Benutzungskennzeichen als Suchkriterien dienen, wenn die hergestellten Querverbindungen bei Entfernung eines Programms aus dem internen Speicher wieder gelöst werden.

3. Verfahren nach Anspruch 1 und 2, dadurch gekennzeichnet, daß bei Entfernung eines Programms aus dem internen Speicher in der zweiten Tabelle des zu entfernenden Programms alle Elemente aufgesucht werden, die ein Benutzungskennzeichen enthalten, daß mit den in diesen Elementen gespeicherten Eingangssymbolen die ersten Tabellen aller anderen im internen Speicher verbleibenden Programme nach gleichnamigen, extern definierten Symbolen durchsucht werden und daß in den ermittelten Elementen der ersten Tabellen die Adresseneintragungen gelöscht werden.

4. Verfahren nach einem der Ansprüche 1 bis 3, dadurch gekennzeichnet, daß die Programmunterbrechung durch einen bzw. eine ein Laden der Adresse eines extern definierten Symbols aus der ersten Tabelle in ein vom Programm benutztes Register steuernden Befehl bzw. Befehlsfolge ausgelöst wird, wenn die zu ladende Adresse den Wert 0 enthält.

5. Verfahren nach einem der Ansprüche 1 bis 4, dadurch gekennzeichnet, daß am Kopf der ersten Tabelle ein Kennzeichen gespeichert wird, das für beide Tabellen je ein Indikatorbit enthält zur Anzeige, ob die betreffende Tabelle im Programm signifikante Eintragungen enthält oder nicht.

6. Verfahren nach Anspruch 5, dadurch gekennzeichnet, daß am Kopf einer jeden Tabelle ein Kennfeld mit der Anzahl der verfügbaren Tabellenelemente gespeichert wird.

7. Verfahren nach einem der Ansprüche 1 bis 6, dadurch gekennzeichnet, daß die im Überwachungsprogramm gebildete gemeinsame Adressentabelle in jedem ihrer Elemente eine Programmidentifizierungsangabe und je eine Adresse der zu dieser Programmidentifizierungsangabe gehörigen ersten und zweiten Tabelle enthält.

8. Verfahren nach Anspruch 7, dadurch gekennzeichnet, daß jedes Element der gemeinsamen Adressentabelle für jede der in ihm enthaltenen Adressen ein Kennfeld aufweist, in welches eine Anzeige eingespeichert wird, wenn in die zugehörige erste Tabelle wenigstens eine externe Adresse eingegeben wurde bzw. wenn die zugehörige zweite Tabelle wenigstens einmal benutzt wurde.

9. Verfahren nach einem der Ansprüche 1 bis 8, dadurch gekennzeichnet, daß die erste und zweite Tabelle am Anfang des Quellenprogramms definiert und beim maschinellen Übersetzen im Objektprogramm gebildet wird und daß für die Markierung der Eingangssymbole und der programmextern definierten Symbole innerhalb des Programms hierfür bereits vorhandene Standardbezeichnungen verwendet werden.

10. Verfahren nach einem der Ansprüche 1 bis 9, dadurch gekennzeichnet, daß beim Laden eines Programms in den internen Speicher zunächst geprüft wird, ob sich die Indikatorbits für das Vorhandensein einer ersten und einer zweiten Tabelle im zu ladenden Programm im 0-Zustand befinden, daß im Verneinungsfalle das nächste freie Element der gemeinsamen Adressentabelle ermittelt und in dessen Programmidentifizierungsfeld der Name dieses Programmes eingeschrieben wird, daß daraufhin jedes der Indikatorbits einzeln geprüft wird, ob es sich im 1-Zustand befindet, und daß im Bejaungsfalle die Adresse der ersten und/oder zweiten Tabelle des Programms in das gleiche Tabellenelement eingespeichert wird.

11. Verfahren nach einem der Ansprüche 1 bis 10, dadurch gekennzeichnet, daß bei der durch den Aufruf eines externen Symbols verursachten Programmunterbrechung das externe Symbol als Suchbegriff gespeichert wird, daß das nächste besetzte Element der gemeinsamen Adressentabelle ermittelt wird, daß geprüft wird, ob dieses Element eine Adresse für eine zweite Tabelle enthält, daß im Bejahungsfalle durch Aufsuchen der Elemente dieser Tabelle geprüft wird, ob eines der dort eingetragenen Eingangssymbole identisch ist mit dem gespeicherten externen Symbol, und daß im Falle einer Identitätsanzeige die im gleichen Element enthaltene, dem Eingangselement zugeordnete Adresse auf das Adressenfeld des externen Symbols in der ersten Tabelle des unterbrochenen Programms übertragen wird.

12. Verfahren nach einem der Ansprüche 1 bis 11, dadurch gekennzeichnet, daß beim Entfernen eines Programms aus dem Hauptspeicher die Elemente mit dem Namen dieses Programms in der gemeinsamen Adressentabelle aufgesucht und gelöscht werden, daß geprüft wird, ob das gefundene Element eine Eintragung im Adressenfeld für die zweite Tabelle enthält und ob die Benutzungsanzeige dieser Adresseneintragung im 1-Zustand ist; daß im Bejahungsfalle beider Prüfungen das Eingangssymbol aus dem betreffenden Element als Suchbegriff gespeichert und die gemeinsame Adressentabelle unter Verwendung eines Laufwertes nach Elementen durchsucht wird, die eine Eintragung im Adressenfeld für die ersten Tabellen enthalten und deren Benutzungsanzeige für diese Adresseneintragung im 1-Zustand ist, daß das externe Symbol eines jeden Elementes der so ermittelten ersten Tabellen mit dem gespeicherten Eingangssymbol auf Identität verglichen wird und daß im Identitätsfalle im gefundenen Element die dem externen Symbol zugeordnete Adresse gelöscht wird.

13. Verfahren nach Anspruch 12, dadurch gekennzeichnet, daß bei fehlender Identität zwischen dem als Suchbegriff gespeicherten Eingangssymbol und dem in der betreffenden ersten Tabelle gefundenen externen Symbol geprüft wird, ob das Element eine Adresseneintragung enthält, daß im Bejahungsfalle ein Anzeigeschalter in den 1-Zustand gebracht und zur Prüfung des nächsten Elementes übergegangen wird, daß nach Prüfung aller Elemente der Tabelle in der gemeinsamen Adressentabelle die Benutzungsanzeige für die durchsuchte erste Tabelle gelöscht wird, wenn der Anzeigeschalter nicht im 1-Zustand steht, und daß danach eine Inkrementierung des Laufwertes zur Weiterschaltung auf das nächste Element der gemeinsamen Adressentabelle erfolgt.

14. Verfahren nach einem der Ansprüche 1 bis 13, dadurch gekennzeichnet, daß für die Aufnahme der ersten und zweiten Tabellen und/oder der gemeinsamen Adressentabelle ein oder mehrere schnelle Arbeitsspeicher verwendet werden.

Leerseite

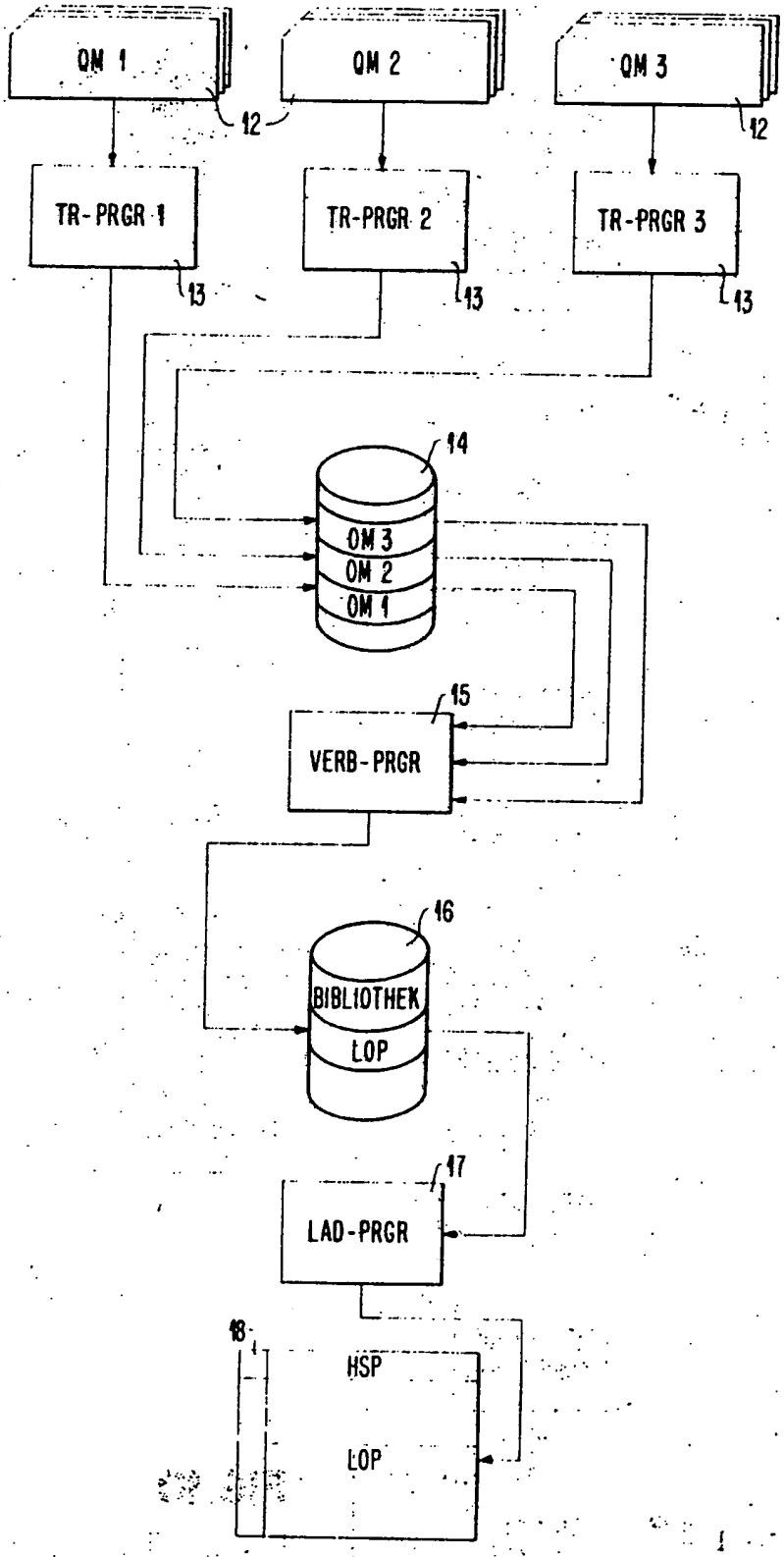


FIG. 1

42 m 3 9-18 AT: 09.10.71 OT: 17.05.73

309820/0325

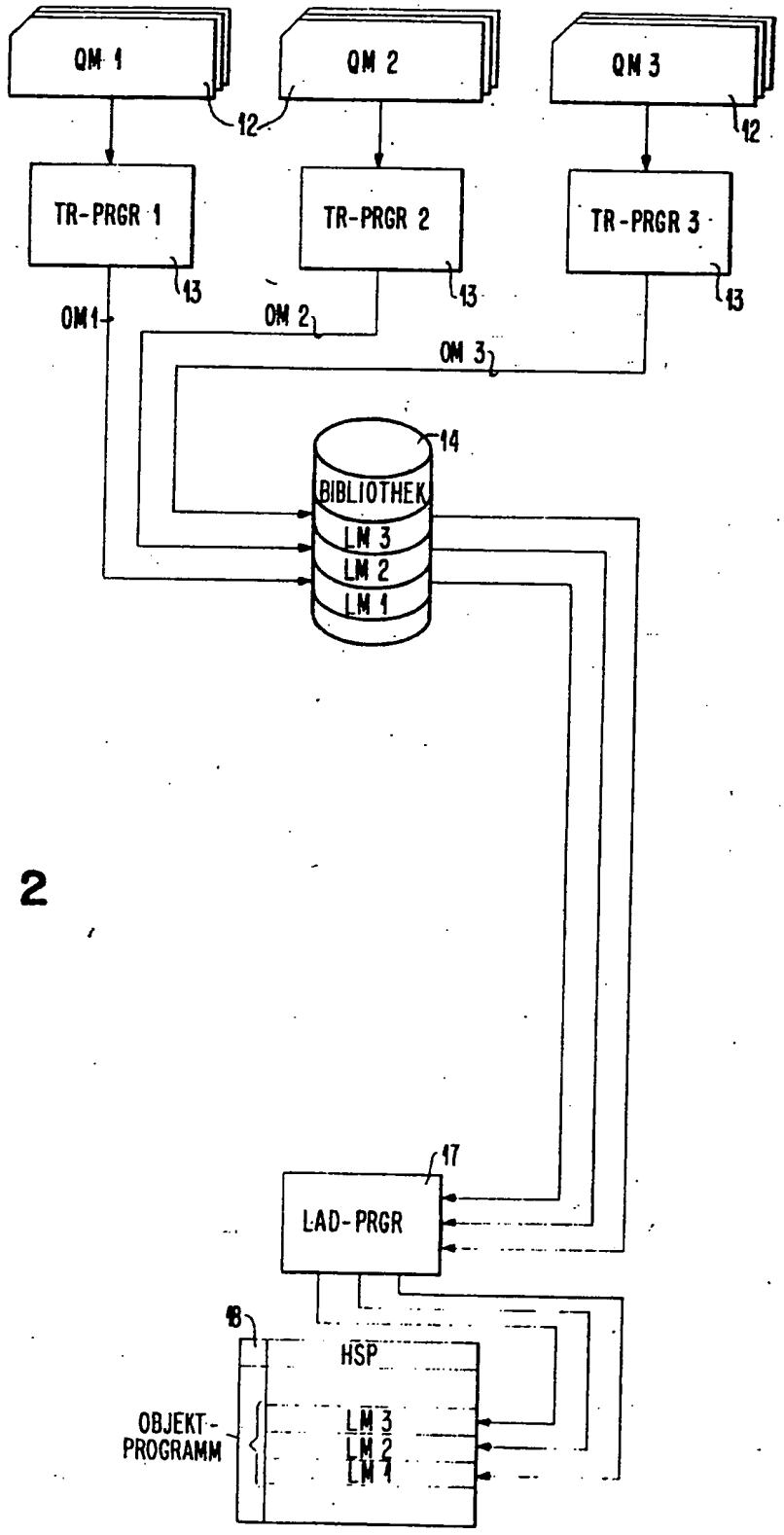

QM 1
QM 2
QM 3
12
12
TR-PRGR 1
TR-PRGR 2
TR-PRGR 3
13
13
13
OM 1
OM 2
OM 3
14
BIBLIOTHEK
LM 3
LM 2
LM 1
17
LAD-PRGR
18
HSP
OBJEKT-
PROGRAMM
LM 3
LM 2
LM 1

FIG. 2

FIG. 3

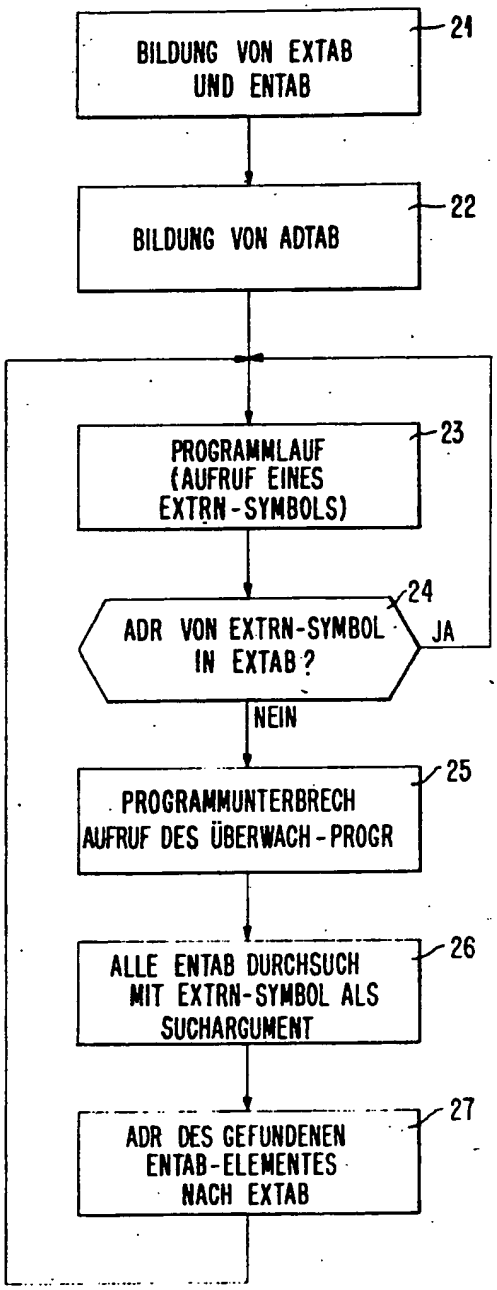

BILDUNG VON EXTAB UND ENTAB
21
BILDUNG VON ADTAB
22
PROGRAMMLAUF (AUFRUF EINES EXTRN-SYMBOLS)
23
ADR VON EXTRN-SYMBOL IN EXTAB ?
24
JA
NEIN
PROGRAMMUNTERBRECH AUFRUF DES ÜBERWACH-PROGR
25
ALLE ENTAB DURCHSUCH MIT EXTRN-SYMBOL ALS SUCHARGUMENT
26
ADR DES GEFUNDENEN ENTAB-ELEMENTES NACH EXTAB
27

FIG. 4A

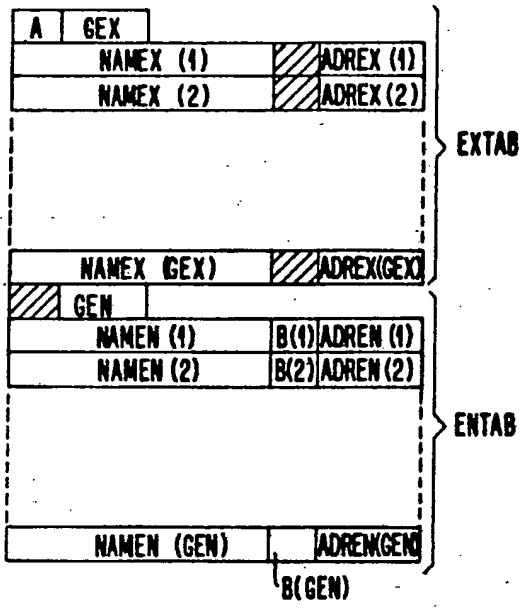


FIG. 4B

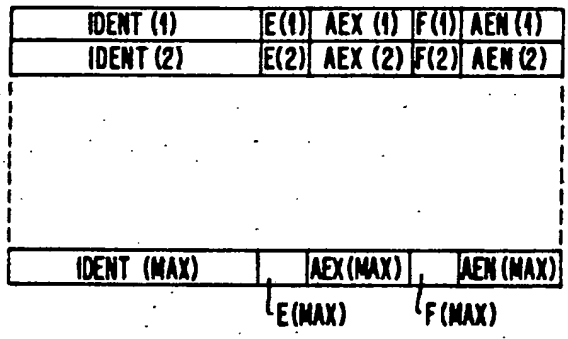


FIG. 5A

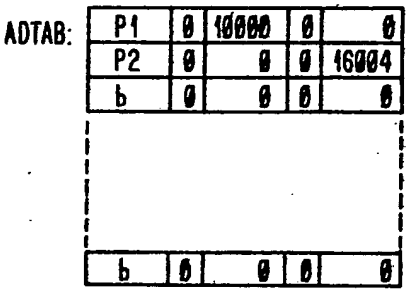


FIG. 5B

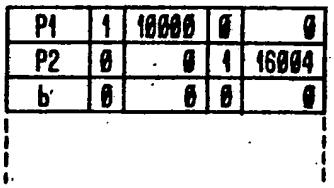
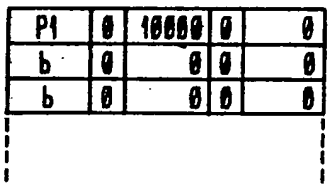
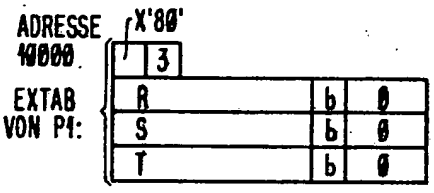
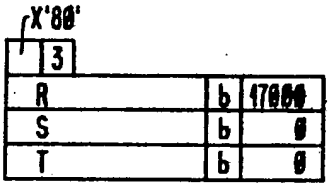
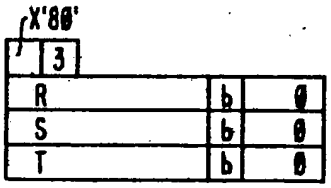


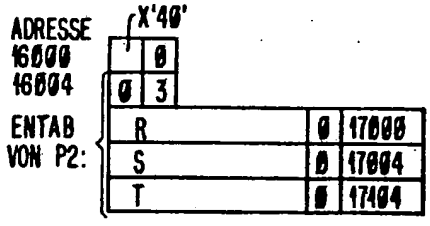
FIG. 5C

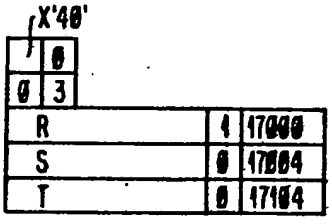












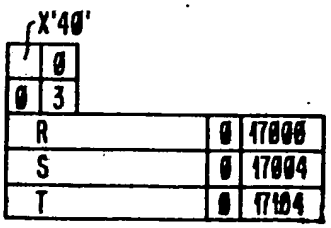


FIG. 6

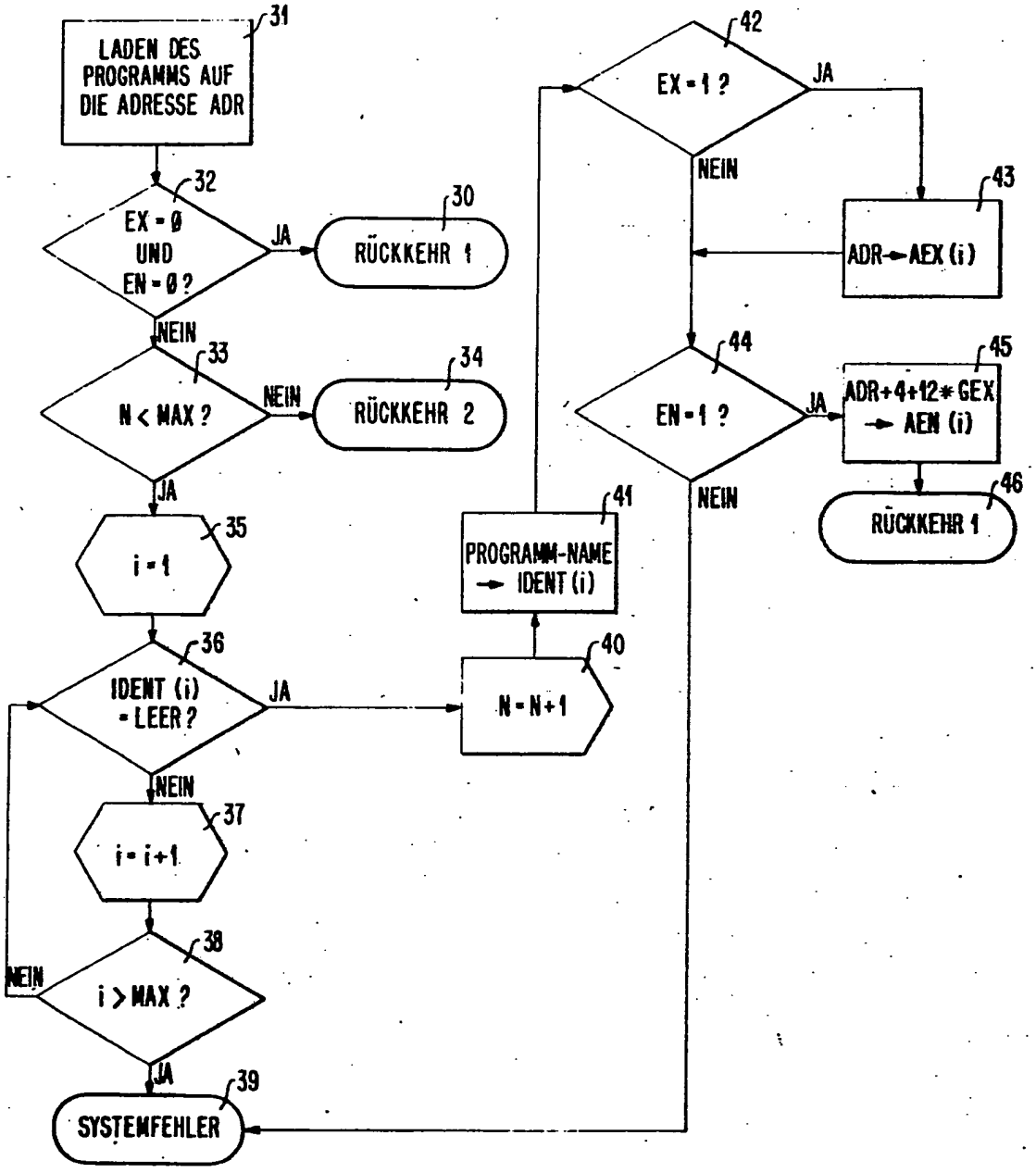

31
LADEN DES PROGRAMMS AUF DIE ADRESSE ADR
32
EX = 0 UND EN = 0 ?
JA
30
RÜCKKEHR 1
NEIN
33
N < MAX ?
NEIN
34
RÜCKKEHR 2
JA
35
i = 1
36
IDENT (i) = LEER ?
JA
NEIN
37
i = i+1
38
i > MAX ?
NEIN
JA
39
SYSTEMFEHLER
40
N = N+1
41
PROGRAMM-NAME → IDENT (i)
42
EX = 1 ?
JA
NEIN
43
ADR → AEX (i)
44
EN = 1 ?
JA
NEIN
45
ADR+4+12*GEX → AEN (i)
46
RÜCKKEHR 1

FIG. 7A

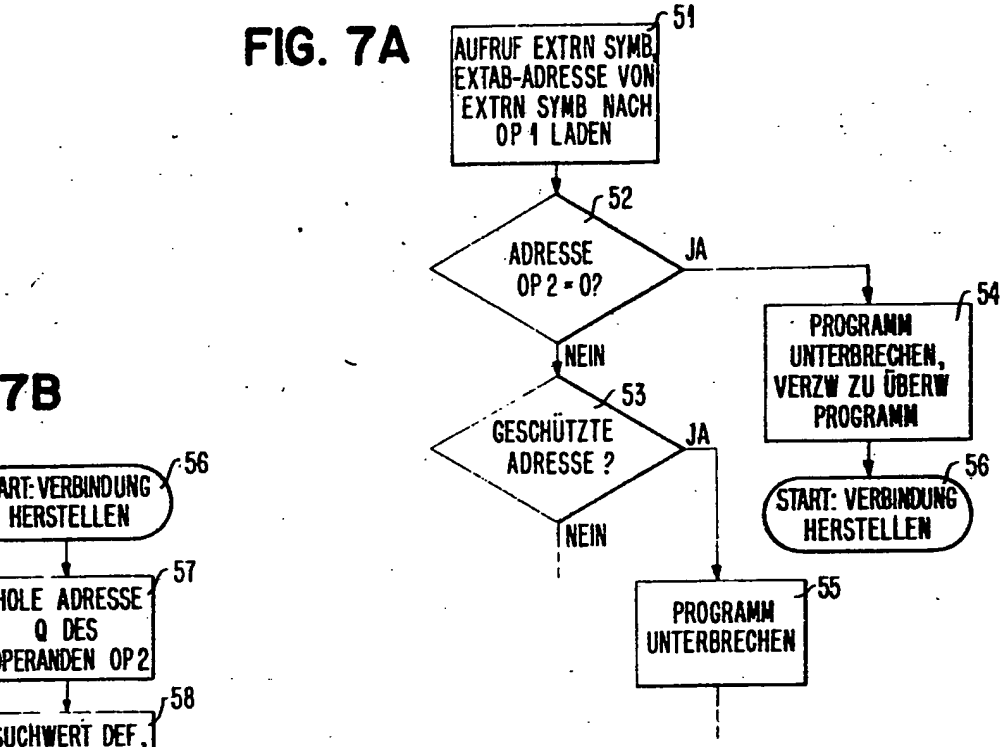
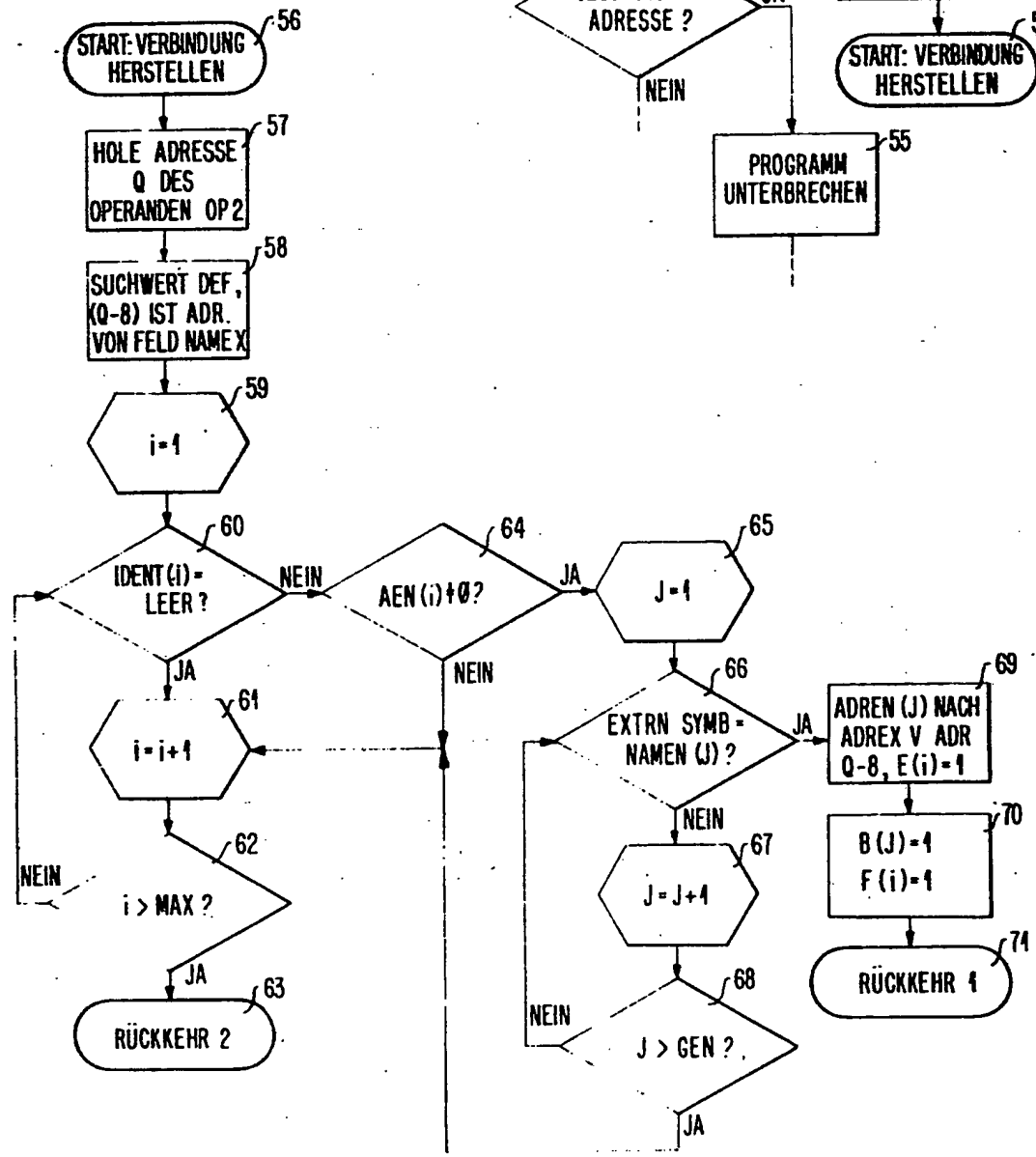

51
AUFRUF EXTRN SYMB, EXTAB-ADRESSE VON EXTRN SYMB NACH OP 1 LADEN
52
ADRESSE OP 2 = 0?
JA
NEIN
54
PROGRAMM UNTERBRECHEN, VERZW ZU ÜBERW PROGRAMM
53
GESCHÜTZTE ADRESSE ?
JA
NEIN
56
START: VERBINDUNG HERSTELLEN
55
PROGRAMM UNTERBRECHEN

FIG. 7B


56
START: VERBINDUNG HERSTELLEN
57
HOLE ADRESSE Q DES OPERANDEN OP 2
58
SUCHWERT DEF, (Q-8) IST ADR. VON FELD NAME X
59
i=1
60
IDENT(i) = LEER ?
NEIN
JA
64
AEN(i)≠0?
JA
NEIN
65
J=1
61
i=i+1
62
i > MAX ?
NEIN
JA
63
RÜCKKEHR 2
66
EXTRN SYMB = NAMEN (J) ?
JA
NEIN
69
ADREN (J) NACH ADREX V ADR Q-8, E(i)=1
70
B(J)=1
F(i)=1
71
RÜCKKEHR 1
67
J=J+1
68
J > GEN ?
NEIN
JA

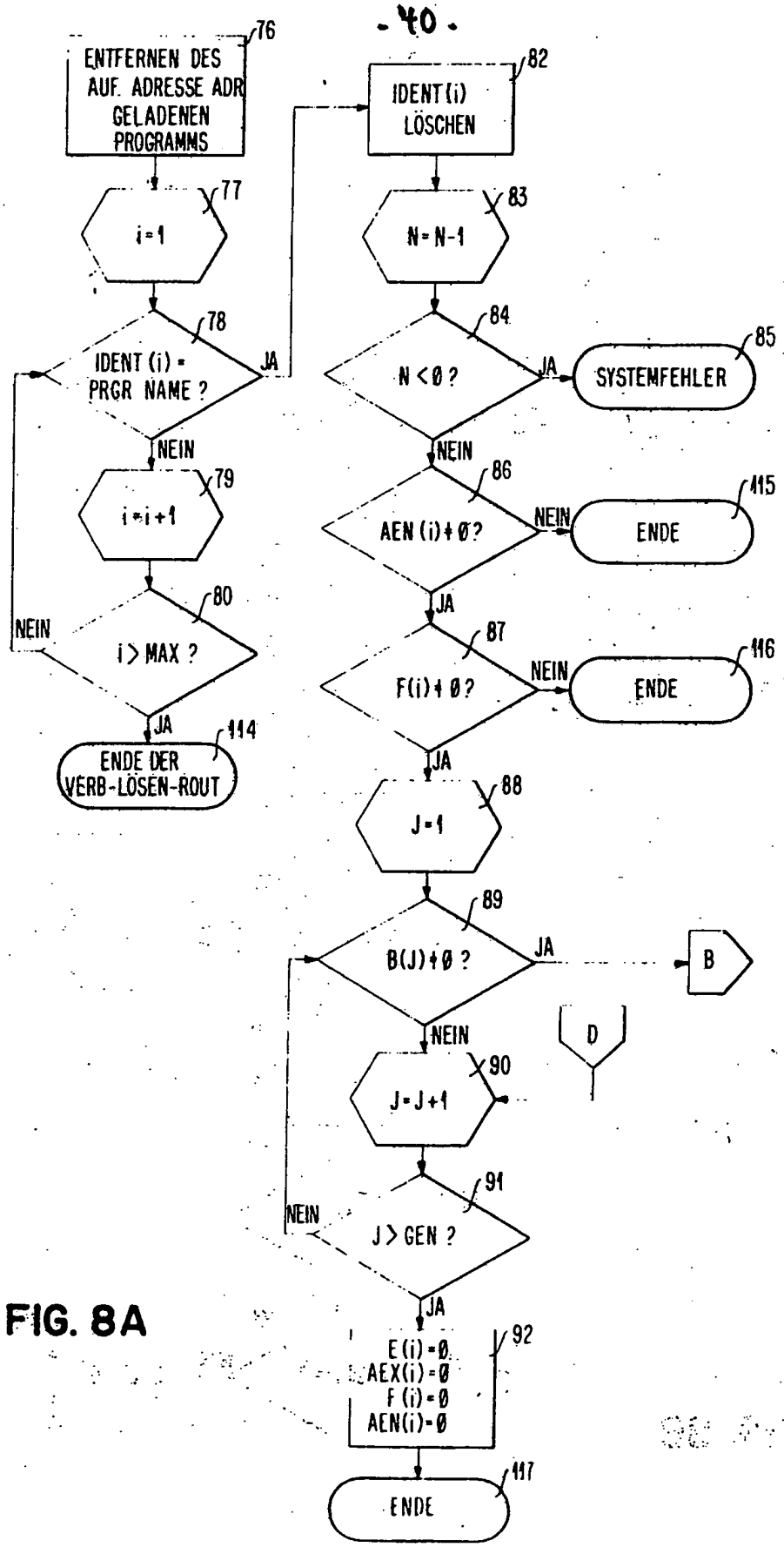

76
ENTFERNEN DES AUF. ADRESSE ADR GELADENEN PROGRAMMS
77
i=1
78
IDENT(i) = PRGR NAME ?
JA
NEIN
79
i=i+1
80
i > MAX ?
NEIN
JA
114
ENDE DER VERB-LÖSEN-ROUT
82
IDENT(i) LÖSCHEN
83
N=N-1
84
N < 0 ?
JA
85
SYSTEMFEHLER
NEIN
86
AEN(i) ≠ 0?
NEIN
115
ENDE
JA
87
F(i) ≠ 0?
NEIN
116
ENDE
JA
88
J=1
89
B(J) ≠ 0 ?
JA
B
NEIN
D
90
J=J+1
91
J > GEN ?
NEIN
JA
92
E(i)=0
AEX(i)=0
F(i)=0
AEN(i)=0
117
ENDE

FIG. 8A

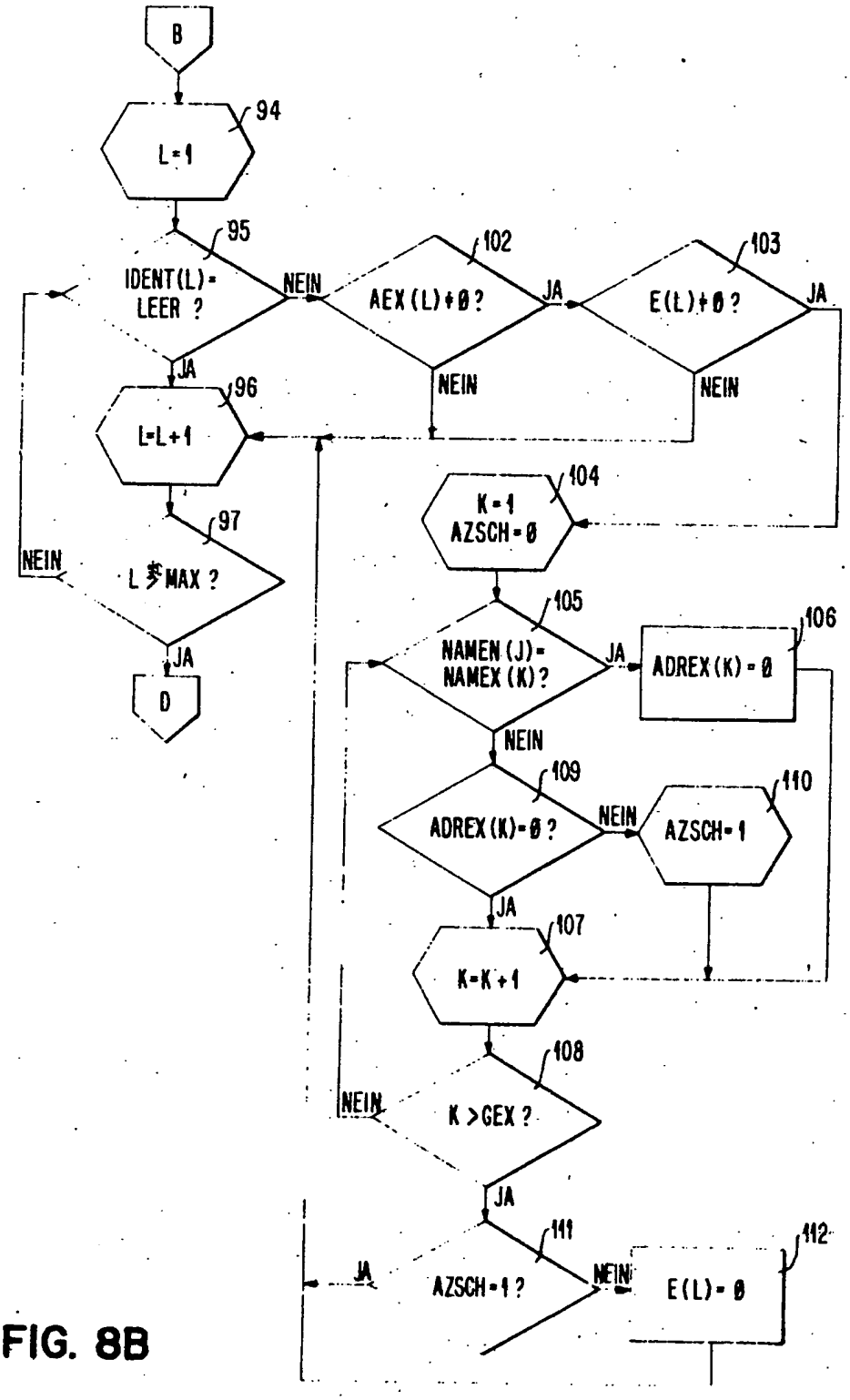

B
94
L=1
95
IDENT(L)= LEER ?
NEIN
JA
102
AEX(L)≠0?
JA
NEIN
103
E(L)≠0?
JA
NEIN
96
L=L+1
97
L ≯ MAX ?
NEIN
JA
D
104
K=1
AZSCH=0
105
NAMEN(J)= NAMEX(K)?
JA
106
ADREX(K)=0
NEIN
109
ADREX(K)=0?
NEIN
110
AZSCH=1
JA
107
K=K+1
108
K > GEX ?
NEIN
JA
111
AZSCH=1?
JA
NEIN
112
E(L)=0

FIG. 8B

FIG. 9

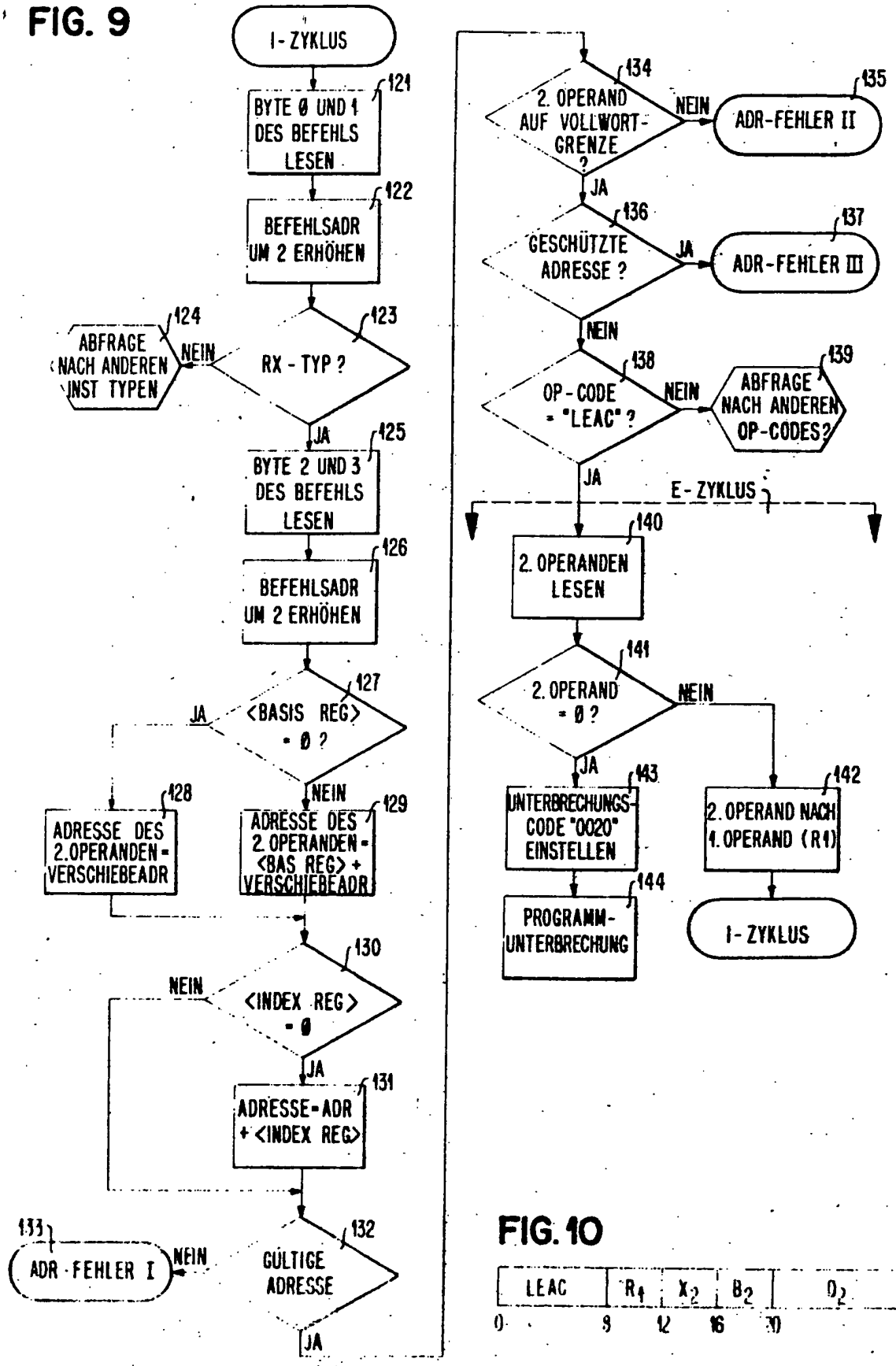

I-ZYKLUS
121
BYTE 0 UND 1 DES BEFEHLS LESEN
122
BEFEHLSADR UM 2 ERHÖHEN
123
RX - TYP ?
NEIN
124
ABFRAGE NACH ANDEREN INST TYPEN
JA
125
BYTE 2 UND 3 DES BEFEHLS LESEN
126
BEFEHLSADR UM 2 ERHÖHEN
127
<BASIS REG> = 0 ?
JA
NEIN
128
ADRESSE DES 2.OPERANDEN-VERSCHIEBEADR
129
ADRESSE DES 2.OPERANDEN = <BAS REG> + VERSCHIEBEADR
130
<INDEX REG> = 0
NEIN
JA
131
ADRESSE = ADR + <INDEX REG>
132
GÜLTIGE ADRESSE
NEIN
133
ADR-FEHLER I
JA
134
2. OPERAND AUF VOLLWORT-GRENZE ?
NEIN
135
ADR-FEHLER II
JA
136
GESCHÜTZTE ADRESSE ?
JA
137
ADR-FEHLER III
NEIN
138
OP-CODE = "LEAC" ?
NEIN
139
ABFRAGE NACH ANDEREN OP-CODES?
JA
E-ZYKLUS
140
2. OPERANDEN LESEN
141
2. OPERAND = 0 ?
NEIN
JA
143
UNTERBRECHUNGS-CODE "0020" EINSTELLEN
144
PROGRAMM-UNTERBRECHUNG
142
2. OPERAND NACH 1. OPERAND (R1)
I-ZYKLUS

FIG. 10

LEAC
R1
X2
B2
D2
0
8
12
16
20
31